L 12

```
   SSSSSSSSSSSS   YYY         YYY     SSSSSSSSSSSS
   SSSSSSSSSSSS   YYY         YYY     SSSSSSSSSSSS
   SSSSSSSSSSSS   YYY         YYY     SSSSSSSSSSSS
SSS               YYY         YYY  SSS
SSS               YYY         YYY  SSS
SSS               YYY         YYY  SSS
SSS                  YYY   YYY     SSS
SSS                  YYY   YYY     SSS
SSS                  YYY   YYY     SSS
   SSSSSSSSS            YYY           SSSSSSSSS
   SSSSSSSSS            YYY           SSSSSSSSS
   SSSSSSSSS            YYY           SSSSSSSSS
            SSS         YYY                    SSS
            SSS         YYY                    SSS
            SSS         YYY                    SSS
            SSS         YYY                    SSS
            SSS         YYY                    SSS
            SSS         YYY                    SSS
SSSSSSSSSSSS            YYY        SSSSSSSSSSSS
SSSSSSSSSSSS            YYY        SSSSSSSSSSSS
SSSSSSSSSSSS            YYY        SSSSSSSSSSSS
```

_$

Ps

YZ

Z$

Z$

Z$

Z$

Z$

Z$

Z$

Z$

Z$

Z$

Z$

Z$

```
  SSSSSSSS  YY      YY   SSSSSSSS    GGGGGGGG  EEEEEEEEEE  TTTTTTTTTT  LL          KK      KK    IIIIII
  SSSSSSSS  YY      YY   SSSSSSSS    GGGGGGGG  EEEEEEEEEE  TTTTTTTTTT  LL          KK      KK    IIIIII
SS          YY      YY SS          GG          EE              TT      LL          KK      KK      II
SS          YY      YY SS          GG          EE              TT      LL          KK      KK      II
SS            YY  YY   SS          GG          EE              TT      LL          KK    KK        II
SS            YY  YY   SS          GG          EE              TT      LL          KK    KK        II
  SSSSSS        YY       SSSSSS    GG          EEEEEEEE        TT      LL          KKKKKK          II
  SSSSSS        YY       SSSSSS    GG          EEEEEEEE        TT      LL          KKKKKK          II
        SS      YY             SS  GG  GGGGGG  EE              TT      LL          KK    KK        II
        SS      YY             SS  GG  GGGGGG  EE              TT      LL          KK    KK        II
        SS      YY             SS  GG      GG  EE              TT      LL          KK      KK      II          ....
        SS      YY             SS  GG      GG  EE              TT      LL          KK      KK      II          ....
SSSSSSSS        YY     SSSSSSSS      GGGGGG    EEEEEEEEEE      TT      LLLLLLLLLL  KK      KK    IIIIII        ....
SSSSSSSS        YY     SSSSSSSS      GGGGGG    EEEEEEEEEE      TT      LLLLLLLLLL  KK      KK    IIIIII        ....
```

```
LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          .TITLE  SYSGETLKI - GET LOCK MANAGER INFORMATION SYSTEM SERVICE
0000     2          .IDENT  'V04-000'
0000     3 ;
0000     4 ;****************************************************************************
0000     5 ;*                                                                          *
0000     6 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000     7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000     8 ;*  ALL RIGHTS RESERVED.                                                    *
0000     9 ;*                                                                          *
0000    10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    11 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
0000    12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    15 ;*  TRANSFERRED.                                                            *
0000    16 ;*                                                                          *
0000    17 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    18 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
0000    19 ;*  CORPORATION.                                                            *
0000    20 ;*                                                                          *
0000    21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    23 ;*                                                                          *
0000    24 ;*                                                                          *
0000    25 ;****************************************************************************
0000    26
0000    27 ;++
0000    28 ; FACILITY: VMS Executive, System services.
0000    29 ;
0000    30 ; ABSTRACT:
0000    31 ;
0000    32 ;       Return system/cluster lock manager information.
0000    33 ;
0000    34 ; ENVIRONMENT: Kernel Mode
0000    35 ;
0000    36 ; AUTHOR: Rod N. Gamache,       CREATION DATE: 15-November-1982
0000    37 ;
0000    38 ; MODIFIED BY:
0000    39 ;
0000    40 ;       V03-014 RNG0014         Rod N. Gamache          3-Aug-1984
0000    41 ;               Make all Lock waiting states map to LKI$C_WAITING.
0000    42 ;
0000    43 ;       V03-013 RNG0013         Rod N. Gamache          24-Jul-1984
0000    44 ;               Stall access to lock database if cluster is re-configuring,
0000    45 ;               call lock manager routine to perform stall operation.
0000    46 ;
0000    47 ;       V03-012 RNG0012         Rod N. Gamache          01-May-1984
0000    48 ;               Restore the PCB address on successive loops through
0000    49 ;               the main proccess code, when doing a wildcard search.
0000    50 ;
0000    51 ;       V03-011 RNG0011         Rod N. Gamache          26-Mar-1984
0000    52 ;               Fix invalid REMLKID that is returned on Local copy LOCKS.
0000    53 ;
0000    54 ;       V03-010 RNG0010         Rod N. Gamache          21-Mar-1984
0000    55 ;               Return correct EPID value, return 2 more longwords in the list
0000    56 ;               items (REMLKID & remCSID). Set size of individual items in
0000    57 ;               list requests.
```

```
0000     58 ;                Return SS$_IVMODE on access mode violations.
0000     59 ;
0000     60 ;        V03-009 CWH3009         CW Hobbs                28-Feb-1984
0000     61 ;                Change IPL synchronization so that $GETLKI can be called
0000     62 ;                at IPL <= IPL$_ASTDEL.  This lets $GETDVI interrogate
0000     63 ;                the XQP's lock value block so that $GETDVI can return
0000     64 ;                the correct value for DVI$_FREEBLOCKS.
0000     65 ;
0000     66 ;        V03-008 RNG0008         Rod N. Gamache          05-Dec-1983
0000     67 ;                Change references to LOCK STRUCTURES to reflect changes made
0000     68 ;                in the Lock Manager.
0000     69 ;
0000     70 ;        V03-007 RNG0007         Rod N. Gamache          07-Oct-1983
0000     71 ;                Fix synchronization problem caused by exec routine that
0000     72 ;                lowers IPL; wrote inline code to replace exec routine.
0000     73 ;
0000     74 ;        V03-006 CWH3006         CW Hobbs                23-Sep-1983
0000     75 ;                Fix broken branch
0000     76 ;
0000     77 ;        V03-005 RNG0005         Rod N. Gamache          31-Aug-1983
0000     78 ;                Deliver AST's only on success.
0000     79 ;                Allow EXEC mode and KERNEL mode users access to system locks.
0000     80 ;                Return zero REMLKID if CSID is zero.
0000     81 ;
0000     82 ;        V03-004 RNG0004         Rod N. Gamache          05-Aug-1983
0000     83 ;                Add REMLKID item code.
0000     84 ;                Return SS$_NOMORELOCK error instead of SS$_NOMOREPROC.
0000     85 ;                Add support for distributed list items (LOCKS, BLOCKEDBY
0000     86 ;                and BLOCKING).
0000     87 ;                Make sure user has sufficient BYCNT quota for list operations.
0000     88 ;                Return proper CSID in the event the CSID of the RSB is zero.
0000     89 ;
0000     90 ;        V03-003 RNG0003         Rod N. Gamache          05-May-1983
0000     91 ;                Return "external" PID wherever necessary. Return
0000     92 ;                SS$_NOWORLD error instead of SS$_NOPRIV.
0000     93 ;                Add support for distributed GETLKI.
0000     94 ;
0000     95 ;        V03-002 SRB0073         Steve Beckhardt         30-Mar-1983
0000     96 ;                Fix broken ASSUME statement.
0000     97 ;
0000     98 ;        V03-001 RNG0001         Rod N. Gamache          14-Mar-1983
0000     99 ;                Remove SYSNAM bit from RMOD field. Change RMOD to be a
0000    100 ;                full byte. Use RMOD in RSB rather than LKB.
0000    101 ;--
```

```
          0000
          0000  ;
          0000  ; EQUATED SYMBOLS:
          0000  ;
          0000
00000002  0000          MAXSTRUC = 2                            ; Maximum number of structures
          0000
00000004  0000          EFN = 4                                 ; event flag number argument
00000008  0000          LKID = 8                                ; address of the lock ID
0000000C  0000          ITMLST = 12                             ; address of item identifiers
00000010  0000          IOSB = 16                               ; I/O status block addrers
00000014  0000          ASTADR = 20                             ; ast routine address
00000018  0000          ASTPRM = 24                             ; ast parameter
0000001C  0000          RESERV = 28                             ; RESERVED
          0000
          0000  ;
          0000  ; One quadword local is left on stack for routines which may
          0000  ; manipulate values before returning them.
          0000  ;
          0000
FFFFFFF8  0000          LOCAL_SPACE = -8
FFFFFFFC  0000          SAVED_IPL   = -4                        ; We will reference stored IPL off the frame
          0000
00000005  0000          MAX_LKB_ITEM = <LKI$_LASTLKB&^XFF>-1 ; maximum LKBTBL item number
0000000B  0000          MAX_RSB_ITEM = <LKI$_LASTRSB&^XFF>-1 ; maximum RSBTBL item number
          0000
          0000  ;
          0000  ; Data type codes (all numeric types have same code)
          0000  ;
          0000
00000000  0000          VALUE = 0                               ; numeric value
00000001  0000          BSTRING = 1                             ; blank filled string
00000002  0000          CSTRING = 2                             ; counted ascii string
          0000  ;
          0000  ; AST control block extensions
          0000  ;
          0000          $DEFINI ACB
          0000
0000001C  0000          .=ACB$L_KAST+4                          ;
          001C
          001C  $DEF    ACB_L_DADDR     .BLKL   1               ; data buffer address
          0020  $DEF    ACB_L_EFN       .BLKL   1               ; event flag number
          0024  $DEF    ACB_L_IOSB      .BLKL   1               ; completion AST routine addr
          0028  $DEF    ACB_L_OPID      .BLKL   1               ; original requester's PID
          002C  $DEF    ACB_L_COUNT     .BLKL   1               ; item descriptor count
          0030  $DEF    ACB_L_ILIST                             ; item descriptor list
          0030
0000000C  0030          ACB_C_IDESC = 12                        ; item descriptor size
          0030
          0030          $DEFEND ACB
          0000
          0000
          0000  ;
          0000  ; OWN STORAGE:
          0000  ;
          0000
      00000000          .PSECT  WSYSGETLKI                      ; Non-paged PSECT
```

```
                     0000   217
                     0000   218
                     0000   219 ;
                     0000   220 ; This array contains the maximum item number for both of the item
                     0000   221 ; data structures, indexed by structure number.
                     0000   222 ;
                     0000   223
                     0000   224 MAXCOUNT:
                  05 0000   225         .BYTE   MAX_LKB_ITEM
                  08 0001   226         .BYTE   MAX_RSB_ITEM
                     0002   227
                     0002   228 ;
                     0002   229 ; The tables contain a word offset followed by a byte code for each item
                     0002   230 ; followed by a byte of structure type.  The code contains the length of
                     0002   231 ; the item in the low five bits, and the item type in the high three bits.
                     0002   232 ; The types are value, counted string, and blank filled string.
                     0002   233 ;
                     0002   234
                     0002   235 LKBTBL:
            00000026 0002   236         .BLKB   6*<MAX_LKB_ITEM+1>              ; define LKB table
                     0026   237 RSBTBL:
            0000005C 0026   238         .BLKB   6*<MAX_RSB_ITEM+1>              ; define RSB table
                     005C   239
                     005C   240         .SAVE                                   ; save current location counter
                     005C   241
                     005C   242 ;
                     005C   243 ; Define entries to LKBTBL
                     005C   244 ;
            00000000 005C   245         LIRSG$K_ZERO = 0                        ; Define empty holder
                     005C   246
                     005C   247         LKBITM  PID,L_EPID,VALUE,4              ; EPID of owner process
                     0008   248         LKBITM  LCKREFCNT,W_REFCNT,VALUE,2      ; sub-lock reference count
                     001A   249         LKBITM  STATE,B_RQMODE,VALUE,3,L_STATE  ; current state of lock
                     000E   250         LKBITM  PARENT,L_PARENT,VALUE,4         ; LOCK ID of parent lock
                     0014   251         LKBITM  LOCKID,L_LKID,VALUE,4           ; LOCK ID of lock
                     0020   252         LKBITM  REMLKID,L_REMLKID,VALUE,4       ; Remote LOCK ID of lock
                     0026   253
                     0026   254 ;
                     0026   255 ; Define entries to RSBTBL
                     0026   256 ;
                     0026   257
                     0026   258         RSBITM  RESNAM,B_RSNLEN,CSTRING,31      ; resource name
                     0032   259         RSBITM  RSBREFCNT,W_REFCNT,VALUE,2,L_RSBREFCNT ; sub-resource reference coun
                     003B   260         RSBITM  VALBLK,Q_VALBLK,BSTRING,16,Q_VALBLK ; value block
                     003E   261         RSBITM  SYSTEM,L_CSID,VALUE,4           ; system id of system which has
                     0044   262                                                 ;  the master copy of resource
                     0044   263         RSBITM  NAMSPACE,W_GROUP,VALUE,4        ; resource name space
                     002C   264         RSBITM  LCKCOUNT,L_GRQFL,VALUE,4,L_LCKCOUNT ; count of locks granted on reso
                     004A   265         RSBITM  BLOCKEDBY,L_GRQFL,VALUE,-       ; list of locks blocked by LKID
                     004A   266                 LKISC_LENGTH
                     0050   267         RSBITM  BLOCKING,L_WTQFL,VALUE,-        ; list of locks blocking LKID
                     0050   268                 LKISC_LENGTH
                     0056   269         RSBITM  LOCKS,L_GRQFL,VALUE,LKISC_LENGTH ; list of associated locks
                     005C   270
                     005C   271
            0000005C 005C   272         .RESTORE                                ; restore location counter
                     005C   273
```

```
                                E 2
005C    274 ;
005C    275 ; Table to define items which must be handled
005C    276 ; by action routines.
005C    277 ;
005C    278
005C    279 SPECIAL:
005C    280             SPECIAL_ITEM    PID,SPC_PID                 ; PID of owner process
0062    281             SPECIAL_ITEM    STATE,SPC_STATE             ; current state of lock
0068    282             SPECIAL_ITEM    PARENT,SPC_PARENT           ; LOCK ID of parent lock
006E    283             SPECIAL_ITEM    SYSTEM,SPC_SYSTEM           ; CSID of master
0074    284             SPECIAL_ITEM    NAMSPACE,SPC_NAMSPACE       ; resource name space
007A    285             SPECIAL_ITEM    LCKCOUNT,SPC_LCKCOUNT       ; count of locks granted on resource
0080    286             SPECIAL_ITEM    BLOCKEDBY,SPC_BLOCKEDBY     ; list of locks blocked by LKID
0086    287             SPECIAL_ITEM    BLOCKING,SPC_BLOCKING       ; list of locks blocking LKID
008C    288             SPECIAL_ITEM    LOCKS,SPC_LOCKS             ; list of associated locks
0092    289             SPECIAL_ITEM    REMLKID,SPC_REMLKID         ; Remote lock id
0098    290
0000000A 0098   291 SPECIAL_LEN = <.-SPECIAL>/6                     ; compute number of entries
```

```
                        .SBTTL  SYSGETLKI - GETLKI get lock manager information system service

;++
;
; FUNCTIONAL DESCRIPTION:
;
;       This service allows a process to receive information about the
;       locks, or any process locks which it has the privilege to examine.
;
; CALLING SEQUENCE:
;
;       CALLS/CALLG
;
;               Actually, this routine MUST be called through the system
;               service dispatcher.
;
; INPUTS:
;
;       R4              PCB address of requesting process
;
;       EFN(AP)         number of the event flag to set when all of the
;                       requested data is valid.
;       LKID(AP)        address of a longword containing the process ID of the
;                       process for which the information is being requested
;       ITMLST(AP)      address of a list of item descriptors of the form:
;
;                               +------------------------------+
;                               !  ITEM CODE   !  BUF. LENGTH !
;                               +------------------------------+
;                               !       BUFFER ADDRESS         !
;                               +------------------------------+
;                               !  ADDRESS TO RETURN LENGTH    !
;                               +------------------------------+
;
;       IOSB(AP)        address of a quadword I/O status block to receive final
;                       status
;       ASTADR(AP)      address of an AST routine to be called when all of the
;                       requested data has been supplied.
;       ASTPRM(AP)      32 bit ast parameter
;
; IMPLICIT INPUTS:
;
;       IPL <= IPL$_ASTDEL      This allows other system services which are
;                               holding mutexes to call $GETLKI.
;
; OUTPUTS:
;
;       none
;
; IMPLICIT OUTPUTS:
;
;       none
;
; ROUTINE VALUE:
;
;       SS$_NORMAL      normal completion.
;       SS$_ACCVIO      ITMLST can not be read by the calling access mode.
```

```
                                                                ;                       or the return buffer or return length word can not
                                                                ;                       be written by the calling access mode.
                                                                ;               SS$_BADPARAM    an invalid item identifier was supplied.
                                                                ;               SS$_IVLOCKID    lock id specified is not valid.
                                                                ;               SS$_BUFFEROVF   data has overflowed the user buffer.
                                                                ;               SS$_NOSYSLCK    SYSLCK privilege is needed to access this information.
                                                                ;               SS$_NOWORLD     WORLD privilege is needed to access this information.
                                                                ;               SS$_EXQUOTA     User's AST or BYTLM quota has been exceeded.
                                                                ;               SS$_INSFMEM     Insufficient non-paged dynamic memory.
                                                                ;               SS$_NOMORELOCK  No more locks on "wildcard" search operation.
                                                                ;
                                                                ; SIDE EFFECTS:
                                                                ;
                                                                ;               none
                                                                ;--
                                                                ;
                                                                .PSECT  YEXEPAGED                       ; Only entry mask in this program section
                                                                ;
                                                                .ENTRY  EXESGETLKI,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                                                                JMP     EXE_GETLKI                      ; Transfer to real procedure
                                                                ;
                                                                .PSECT  WSYSGETLKI
                                                                ;
                                                                EXE_GETLKI:
                                                                DSBINT  #IPL$_SYNCH                     ; Raise IPL to check lock mgr database
                                                                JSB     G^LCK$CHECK_STALL               ;  ... stall if needed (in CALLER's mode)
                                                                SETIPL  #IPL$_ASTDEC                    ; Set IPL to ASTDEL
                                                                MOVAL   LOCAL_SPACE(SP),SP              ; Allocate local space on stack
                                                                CLRL    R11                             ; Assume no remote LOCK information
                                                                MOVL    SCH$GL_CURPCB,R4                ; Reset PCB address
                                                                MOVL    R11,R0                          ; Any remote LOCK BLOCK?
                                                                BEQL    5$                              ; Br if not, okay
                                                                CLRL    R11                             ; No more remote lock block
                                                                JSB     EXE$DEANONPAGED                 ; Else, deallocate the remote lock block
                                                                BSBW    GETLKB                          ; Get LKB address of desired lock
                                                                BLBC    R0,17$                          ; Exit if invalid LKID specified
                                                                ;
                                                                ; Check for, and clear event flag
                                                                ;
                                                                MOVZBL  EFN(AP),R3                      ; Get event flag number
                                                                JSB     SCH$CLREF                       ; Clear this event flag
                                                                BLBC    R0,17$                          ; And return on errors.
                                                                ;
                                                                ; Check for, and clear possible IOSB
                                                                ;
                                                                MOVL    IOSB(AP),R1                     ; Get IOSB address
                                                                BEQL    5$                              ; Branch if none
                                                                IFWRT   #8,(R1),4$                      ; Check access to it
                                                                BRW     30$                             ; Else, return error
                                                                4$:     CLRQ    (R1)                    ; Clear IOSB
                                                                ;
                                                                ; Validate AST, if present. Note R4 still has our PCB address, and R9
                                                                ; has the LKB address of the lock we want information from.
                                                                ;
                                                                5$:     TSTL    ASTADR(AP)
                                                                BEQL    7$                              ; No AST to check.
                                                                TSTW    PCB$W_ASTCNT(R4)                ; Is quota exceeded?
```

```
                 7B   15   00ED   407          BLEQ    35$                     ; Branch if so and return error
                           00EF   408 ;
                           00EF   409 ; Check if information is contained on another system in the cluster
                           00EF   410 ;
                    00DE   30  00EF   411 7$:       BSBW    GET_REMLKI              ; Get remote LKI block if needed
                  2E   50   E9  00F2   412          BLBC    R0,17$                  ; Exit on error
                           00F5   413 ;
                           00F5   414 ; Loop through the item descriptor blocks, validating the requested item
                           00F5   415 ; identifiers and moving accessible items.  A zero item identifier terminates
                           00F5   416 ; the list.
                           00F5   417 ;
                           00F5   418 ; At this point:
                           00F5   419 ;
                           00F5   420 ;       R4 = PCB address
                           00F5   421 ;       R9 = LKB address
                           00F5   422 ;       R11 = Remote lock block information or zero
                           00F5   423 ;       AP = Pointer to argument list
                           00F5   424 ;
              55   0C   AC   D0  00F5   425 10$:       MOVL    ITMLST(AP),R5           ; Get item descriptor list address
                           00F9   426          IFNORD  #4,(R5),30$             ; Check first longword readable
                 56   85   3C  00FF   427 15$:       MOVZWL  (R5)+,R6                ; Get buffer size
                 51   85   3C  0102   428          MOVZWL  (R5)+,R1                ; Get item identifier
                      6D   13  0105   429          BEQL    50$                     ; Done if zero, take normal exit
                           0107   430          IFNORD  #12,(R5),30$            ; Check rest of this descriptor ...
                           010D   431 ;                                          ; ... plus first longword of next one
                 57   85   7D  010D   432          MOVQ    (R5)+,R7                ; Get buffer address and return address
                      51   DD  0110   433          PUSHL   R1                      ; Save R1 across accessibility check
                 50   57   D0  0112   434          MOVL    R7,R0                   ; Buffer address to R0
                 51   56   D0  0115   435          MOVL    R6,R1                   ; And size to R1
                      53   D4  0118   436          CLRL    R3                      ; PROBE will use PSL<PRVMOD>
          00000000'EF        16  011A   437          JSB     EXE$PROBEW              ; Check write accessibility of buffer
                      51 8ED0  0120   438          POPL    R1                      ; Restore R1 for use by CHECKITEM
                51   50   E9  0123   439 17$:       BLBC    R0,GRET                 ; Return error if inaccessible
                           0126   440 ;
                           0126   441 ; We will raise IPL to IPL$_SYNCH to lock down the LKB. We will
                           0126   442 ; have to verify that the LKB is still valid, before proceeding.
                           0126   443 ;
                           0126   444 ; The IPL will be restored by the MOVEIT routine just before copying
                           0126   445 ; the data to the users's buffer. This is done to allow the SPC_xxx
                           0126   446 ; routines to gather up any additional information that needs to be
                           0126   447 ; returned to the user, without verifying that the LKB address is
                           0126   448 ; still valid.
                           0126   449 ;
                           0126   450          $SETIPL #IPL$_SYNCH             ; Raise IPL to sync access to structures
                           0129   451 ;                                          ;  can't reference user's process space
              54   30   A9   3C  0129   452          MOVZWL  LKB$L_LKID(R9),R4       ; Get lock index
                 54   01   CD  012D   453          CMPL    R4,LCK$GL_MAXID         ; Is the lock index still ok?
                      0A   1A  0134   454          BGTRU   20$                     ; Br if no, check for error condition
          00000000'FF44   59   D1  0136   455          CMPL    R9,@LCK$GL_IDTBL[R4]    ; Is the lock address still the same?
                      12   13  013E   456          BEQL    25$                     ; Br if yes, okay to proceed
                           0140   457 20$:       $SETIPL #IPL$_ASTDEL            ; Restore the IPL on error condition
                    08   BC   D5  0143   458          TSTL    @LKID(AP)               ; Is this a "wildcard" search?
                         03   14  0146   459          BGTR    23$                     ; Br if no, continue
                      FF62   31  0148   460          BRW     25$                     ; Else, try for next lock
              50   2124   8F   3C  014B   461 23$:       MOVZWL  #SS$_IVLOCKID,R0        ; Invalid lock id
                      25   11  0150   462          BRB     GRET                    ; Return to user
                           0152   463 ;
```

```
                                0152   464          ; Check item code and return the info to user.
                                0152   465          ;
                          55    DD    0152   466 25$:    PUSHL   R5                      ; Save R5 from action routines
                        009C    30    0154   467          BSBW    CHECKITEM               ; Validate identifier and get item info.
                     1F 50      E9    0157   468          BLBC    R0,40$                  ; Invalid item if error
                        00FD    30    015A   469          BSBW    MOVEIT                  ; Move item to user
                                      015D   470                                          ;  NOTE: IPL is restored to IPL$_ASTDEL
                          55  8ED0   015D   471          POPL    R5                      ; Restore R5
                       9C 50  E8    0160   472          BLBS    R0,15$                  ; Back for next descriptor if ok
                          12  11    0163   473          BRB     GRET                    ; Else, return error
                                0165   474
                    50    0C  3C    0165   475 30$:    MOVZWL  #SS$_ACCVIO,R0          ; Access violation
                          00  11    0168   476          BRB     GRET
                                016A   477
                    50    1C  3C    016A   478 35$:    MOVZWL  #SS$_EXQUOTA,R0         ; AST quota exceeded
                          08  11    016D   479          BRB     GRET
                                016F   480
                    50    14  3C    016F   481 40$:    MOVZWL  #SS$_BADPARAM,R0        ; Illegal item or request
                          03  11    0172   482          BRB     GRET
                                0174   483
                    50    01  3C    0174   484 50$:    MOVZWL  #SS$_NORMAL,R0          ; Normal return
                                0177   485
                                0177   486 ;
                                0177   487 ; Set the event flag, post the completion status, and declare a completion AST
                                0177   488 ;
                          50    DD    0177   489 GRET:   PUSHL   R0                      ; Save completion status
                    50    58  D0    0179   490          MOVL    R11,R0                  ; Any remote lock block?
                          06  13    017C   491          BEQL    5$                      ; Br if not, okay
                00000000'EF   16    017E   492          JSB     EXE$DEANONPAGED         ; Else, deallocate the remote lock block
                                0184   493 5$:     SETIPL  SAVED_IPL(FP)           ; Restore IPL to that on entry to service
                54 00000000'EF  D0    0188   494          MOVL    SCH$GL_CURPCB,R4        ; Get PCB address
                   51    60  A4    D0    018F   495          MOVL    PCB$L_PID(R4),R1        ; Get process's PID
                               52    D4    0193   496          CLRL    R2                      ; Set null priority increment
                   53    04  AC    D0    0195   497          MOVL    EFN(AP),R3              ; Get event flag number to set
                00000000'EF       16    0199   498          JSB     SCH$POSTEF              ; Set the event flag
                   51    10  AC    D0    019F   499 10$:    MOVL    IOSB(AP),R1             ; Get address of IOSB
                               09    13    01A3   500          BEQL    20$                     ; Branch if none
                                      01A5   501                                          ; Check if writable
                         61    6E    D0    01A8   502          MOVL    (SP),(R1)               ; Store completion status
                      55    14  AC    D0    01AB   503 20$:    MOVL    ASTADR(AP),R5           ; Get address of AST routine
                               18    13    01AE   504          BEQL    30$                     ; Branch if none specified
                            15 6E    E9    01B0   505          BLBC    (SP),30$                ; No completion AST on error!
                               54    DC    01B3   506          MOVPSL  R4                      ; Get PSL
                54    54    02    16    EF    01B5   507          EXTZV   #PSL$V_PRVMOD,#PSL$S_PRVMOD,R4,R4 ; Extract previous mode
                                      01BA   508          $DCLAST_S (R5),ASTPRM(AP),R4    ; Queue the completion AST
                                50  8ED0   01BF   509 30$:    POPL    R0                      ; Restore completion status
                                      04    01C1   510          RET                     ; And return.
                                      01C2   511
```

```
                        ;  2
                        .SBTTL  GET_REMLKI- Get remote LKI block
                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       Routine to get the remote LKI block if necessary.
                ;
                ; CALLING SEQUENCE:
                ;
                ;       JSB/BSB
                ;
                ; INPUTS:
                ;
                ;       R4      PCB address
                ;       R9      LKB address
                ;       R11     ZERO
                ;
                ; IMPLICIT INPUTS:
                ;
                ;       IPL = IPL$_ASTDEL
                ;
                ; OUTPUTS:
                ;
                ;       R0      success/failure of operation + special flags
                ;       R4      PCB address
                ;       R9      LKB address
                ;       R11     Address of remote LKI block or zero
                ;
                ; IMPLICIT OUTPUTS:
                ;
                ;       none
                ;
                ; SIDE EFFECTS:
                ;
                ;       R0-R3,R8 destroyed.
                ;--
                ;
                        .ENABL  LSB
                GET_REMLKI:                             ; Get remote LKI block
50  01  90      01D0        MOVB    #1,R0               ; Assume success
    04  E0      01D3        BBS     #LKB$V_MSTCPY,-     ; Br if this is the master copy,
1A  2A  A9                  01D5        LKB$W_STATUS(R9),10$    ;  information is local to this system
58  50  A9  D0 01D8        MOVL    LKB$L_RSB(R9),R8    ; Get RSB address
53  38  A8  D0 01DC        MOVL    RSB$L_CSID(R8),R3   ; Is this a process copy?
        10  13  01E0        BEQL    10$                 ; Br if not, information is still local
                01E2        SETIPL  #IPL$_SYNCH         ; Raise IPL to SYNCH
    00000000'GF 16  01E5        JSB     G^LKI$SND_STDREQ    ; And send request for information
                01EB                                    ;  to remote system
54  00000000'EF D0  01EB        MOVL    SCH$GL_CURPCB,R4    ; Get our PCB address
        05  01F2    163 10$:    RSB                         ; Return to caller
                01F3    164
                01F3    165         .DSABL  LSB
                01F3    166
```

```
                                                .SBTTL  CHECKITEM - Validate item identifier

                                ;++
                                ; FUNCTIONAL DESCRIPTION:
                                ;
                                ;       Routine to validate item identifier and return information
                                ;       about the item.
                                ;
                                ; CALLING SEQUENCE:
                                ;
                                ;       JSB/BSB
                                ;
                                ; INPUTS:
                                ;
                                ;       R1      item identifier
                                ;       R9      LKB address
                                ;       R11     REMOTE LKI BLOCK or zero
                                ;
                                ; IMPLICIT INPUTS:
                                ;
                                ;       IPL = IPL$_SYNCH
                                ;
                                ; OUTPUTS:
                                ;
                                ;       R0      success/failure of operation + special flags
                                ;       R1      item identifier
                                ;       R2      structure number
                                ;       R3      item length
                                ;       R4      item address
                                ;       R5      item type code
                                ;
                                ; IMPLICIT OUTPUTS:
                                ;
                                ;       none
                                ;
                                ; SIDE EFFECTS:
                                ;
                                ;       none
                                ;--
                                ;
                                CHECKITEM:
                 50     D4      CLRL    R0                      ; Assume bad item code
                 53     51  9A  MOVZBL  R1,R3                   ; Get item number
          52     51     08  08  EF  EXTZV   #8,#8,R1,R2         ; Get structure number
                        5A  13  BEQL    80$                     ; Error if structure number zero
                 02     52  91  CMPB    R2,#MAXSTRUC            ; Structure number valid?
                        55  1A  BGTRU   80$                     ; Error if not
              FDF5 CF42 53  91  CMPB    R3,MAXCOUNT-1[R2]       ; Check max item values (1 origin)
                        4D  1A  BGTRU   80$                     ; Error if illegal item number
                                CASE    R2,<10$,30$>-B,#1       ; Case on structure base
                                ;
                                ; LKB return item
                                ;
                 54     59  D0  MOVL    R9,R4                   ; Get back LKB address
           55    FDE7 CF  DE  MOVAL   LKBTBL,R5               ; Get address of LKB item table
                        09  11  BRB     40$                     ; Continue
```

```
                     021E  625
                     021E  626     ;
                     021E  627     ; RSB return item
                     021E  628     ;
                     021E  629 50$:  MOVL    LKB$L_RSB(R9),R4        ; Get resource block address
                     0222  630     MOVAL   RSB$TBL,R5              ; Get address of PHD item table
                     0227  631     ;
                     0227  632 40$:  ASHL    #1,R3,R0                ; Double item number
                     022B  633     MOVAL   (R5)[R3],R3             ; Compute address in item table
                     022F  634     ADDL    R0,R3                   ;
                     0232  635     MOVZWL  (R3)+,R5                ; Get offset into data structure
                     0235  636     ADDL    R5,R4                   ; Form complete address
                     0238  637     MOVL    #1,R0                   ; Set successful return
                     023B  638     TSTL    R11                     ; Is there a remote LKI block?
                     023D  639     BEQL    50$                     ; Br if not, continue
                     023F  640     TSTW    2(R3)                   ; Is this item in remote LKI block?
                     0242  641     BEQL    50$                     ; Br if not
                     0244  642     MOVZWL  2(R3),R4                ; Else, get offset in remote LKI block
                     0248  643     ADDL    R11,R4                  ; Form complete address
                     024B  644     BBSS    #1,R0,50$               ; Indicate that no special lookup needed
                     024F  645 50$:  EXTZV   #5,#3,(R3),R5           ; Get item type code
                     0254  646     EXTZV   #0,#5,(R3),R3           ; Get item length
                     0259  647 80$:  RSB                             ; Return to caller
                     025A  648
```

```
                                .SBTTL  MOVEIT - Move data to user's buffer

                        ;++
                        ; FUNCTIONAL DESCRIPTION:
                        ;
                        ;       Move the requested data to user buffer. Zero fill to end of buffer.
                        ;       Return actual data length to user.  Assumes user's buffer has
                        ;       been probed.
                        ;
                        ; CALLING SEQUENCE:
                        ;
                        ;       JSB/BSB
                        ;
                        ; INPUTS:
                        ;
                        ;       R0      special lookup flag
                        ;       R1      item identifier
                        ;       R2      data structure number
                        ;       R3      item length
                        ;       R4      item address
                        ;       R5      item type code
                        ;       R6      user buffer length
                        ;       R7      user buffer address
                        ;       R8      address to return length
                        ;       R9      LKB address
                        ;
                        ; IMPLICIT INPUTS:
                        ;
                        ;       IPL = IPL$_SYNCH
                        ;
                        ; OUTPUTS:
                        ;
                        ;       none
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;
                        ;       IPL = IPL$_ASTDEL
                        ;
                        ; ROUTINE VALUE:
                        ;
                        ;       SS$_NORMAL      Normal successful completion
                        ;       SS$_ACCVIO      Access violation on attempt to access return size
                        ;
                        ; SIDE EFFECTS:
                        ;
                        ;       Registers R1-R4 destroyed
                        ;--
                        ;
                        MOVEIT:
                        ;
                        ; Call routine to check for special conditions
                        ;
                        ;
5A  D4  025A  702               CLRL    R10                     ; No buffer to deallocate - yet!
02 50   01  E0  025C  703       BBS     #1,R0,5$                ; Br if no special lookup needed
                48  10  0260  704       BSBB    CHECK_SPC               ; Check for special actions
                        0262  705 5$:   SETIPL  #IPL$_ASTDEL            ; Restore IPL to ASTDEL
```

```
                     2E 50   E9  0265   706          BLBC    R0,40$                  ; Br if error
                                0268   707          ;
                                0268   708          ; Check for counted string, and find actual length if so.
                                0268   709          ;
                     55 02   D1  0268   710          CMPL    #CSTRING,R5             ; Is this special string?
                        03   12  026B   711          BNEQ    10$                     ; Br if not
                     53 84   9A  026D   712          MOVZBL  (R4)+,R3                ; Get length and skip length byte
                                0270   713          ;
                                0270   714          ; Move the data
                                0270   715          ;
                        28   BB  0270   716  10$:    PUSHR   #^M<R3,R5>              ; Save registers
 67   56   00   64         2C  0272   717          MOVC5   R3,(R4),#0,R6,(R7)      ; Move data to user's buffer, zero fill
                        28   BA  0278   718          POPR    #^M<R3,R5>              ; Restore registers
                        58   D5  027A   719          TSTL    R8                      ; Did caller want return length?
                        15   13  027C   720          BEQL    30$                     ; Br if not
                             7E  027E   721          IFNOWRT #4,(R8),70$             ; Br if longword not writeable
                     56 53   B1  0284   722          CMPW    R3,R6                   ; See how much was moved
                        07   15  0287   723          BLEQ    20$                     ; Use valid data length if it fits
                     53 56   B0  0289   724          MOVW    R6,R3                   ; Else give him "too short" buffer size
                  00 53   1F   E2  028C   725          BBSS    #31,R3,20$              ; And return buffer overflow indicator
                     68 53   D0  0290   726  20$:    MOVL    R3,(R8)                 ; Return length to user
                     50 01   9A  0293   727  30$:    MOVZBL  S^#SS$_NORMAL,R0        ; Set success code
                        5A   D5  0296   728  40$:    TSTL    R10                     ; Any pool deallocation needed?
                        0D   13  0298   729          BEQL    50$                     ; Br if no
                        0F   BB  029A   730          PUSHR   #^M<R0,R1,R2,R3>        ; Save registers
                     50 5A   D0  029C   731          MOVL    R10,R0                  ; Get buffer address
            00000000'EF 16   FB  029F   732          JSB     EXE$DEANONPAGED         ; Deallocate the pool
                        0F   BA  02A5   733          POPR    #^M<R0,R1,R2,R3>        ; Save registers
                        05   05  02A7   734  50$:    RSB                             ; Return to caller
                                02A8   735          ;
                     50 0C   3C  02A8   736  70$:    MOVZWL  #SS$_ACCVIO,R0          ; Return error code
                        E9   11  02AB   737          BRB     40$                     ; Return to caller
                                02AD   738
```

```
                                                     N 2
                     2E 50    E9  0265   706          BLBC    R0,40$                  ; Br if error
                                  0268   707          ;
                                  0268   708          ; Check for counted string, and find actual length if so.
                                  0268   709          ;
                  55 02       D1  0268   710          CMPL    #CSTRING,R5             ; Is this special string?
                     03       12  026B   711          BNEQ    10$                     ; Br if not
                  53 84       9A  026D   712          MOVZBL  (R4)+,R3                ; Get length and skip length byte
                                  0270   713          ;
                                  0270   714          ; Move the data
                                  0270   715          ;
                     28       BB  0270   716 10$:     PUSHR   #^M<R3,R5>              ; Save registers
67 56 00 64          53       2C  0272   717          MOVC5   R3,(R4),#0,R6,(R7)      ; Move data to user's buffer, zero fill
                     28       BA  0278   718          POPR    #^M<R3,R5>              ; Restore registers
                     58       D5  027A   719          TSTL    R8                      ; Did caller want return length?
                     15       13  027C   720          BEQL    30$                     ; Br if not
                                  027E   721          IFNOWRT #4,(R8),70$             ; Br if longword not writeable
                  56 53       B1  0284   722          CMPW    R3,R6                   ; See how much was moved
                     07       15  0287   723          BLEQ    20$                     ; Use valid data length if it fits
                  53 56       B0  0289   724          MOVW    R6,R3                   ; Else give him "too short" buffer size
               00 53 1F       E2  028C   725          BBSS    #31,R3,20$              ; And return buffer overflow indicator
                  68 53       D0  0290   726 20$:     MOVL    R3,(R8)                 ; Return length to user
                  50 01       9A  0293   727 30$:     MOVZBL  S^#SS$_NORMAL,R0        ; Set success code
                     5A       D5  0296   728 40$:     TSTL    R10                     ; Any pool deallocation needed?
                     0D       13  0298   729          BEQL    50$                     ; Br if no
                     0F       BB  029A   730          PUSHR   #^M<R0,R1,R2,R3>        ; Save registers
                  50 5A       D0  029C   731          MOVL    R10,R0                  ; Get buffer address
            00000000'EF       16  029F   732          JSB     EXE$DEANONPAGED         ; Deallocate the pool
                     0F       BA  02A5   733          POPR    #^M<R0,R1,R2,R3>        ; Save registers
                              05  02A7   734 50$:     RSB                             ; Return to caller
                                  02A8   735          ;
                  50 0C       3C  02A8   736 70$:     MOVZWL  #SS$_ACCVIO,R0          ; Return error code
                     E9       11  02AB   737          BRB     40$                     ; Return to caller
                                  02AD   738
```

```
                        02AD   740          .SBTTL  SPECIAL - Handle special conditions
                        02AD   741
                        02AD   742 ;++
                        02AD   743 ;
                        02AD   744 ; FUNCTIONAL DESCRIPTION:
                        02AD   745 ;
                        02AD   746 ;       These routines handle data items which must be transformed
                        02AD   747 ;       before they are returned to the user.  Generally, some
                        02AD   748 ;       transformation is applied to the data item and the newly
                        02AD   749 ;       computed item is stored in LOCAL_SPACE on the stack.
                        02AD   750 ;       The handling routine then changes R4 to point to LOCAL_SPACE
                        02AD   751 ;       so that MOVEIT will move the item from local storage.
                        02AD   752 ;
                        02AD   753 ; CALLING SEQUENCE:
                        02AD   754 ;
                        02AD   755 ;       JSB/BSB
                        02AD   756 ;
                        02AD   757 ; INPUTS:
                        02AD   758 ;
                        02AD   759 ;       R1      item identifier
                        02AD   760 ;       R3      item length
                        02AD   761 ;       R4      item address
                        02AD   762 ;       R6      user buffer length
                        02AD   763 ;       R9      LKB address
                        02AD   764 ;       R10     zero
                        02AD   765 ;
                        02AD   766 ; IMPLICIT INPUTS:
                        02AD   767 ;
                        02AD   768 ;       IPL = IPL$_SYNCH
                        02AD   769 ;
                        02AD   770 ; OUTPUTS:
                        02AD   771 ;
                        02AD   772 ;       R10     system buffer address to deallocate or zero if none
                        02AD   773 ;
                        02AD   774 ; IMPLICIT OUTPUTS:
                        02AD   775 ;
                        02AD   776 ;       none
                        02AD   777 ;
                        02AD   778 ; ROUTINE VALUE:
                        02AD   779 ;
                        02AD   780 ;       SS$_NORMAL      Normal successful completion
                        02AD   781 ;       SS$_INSFMEM     Insufficient non-paged dynamic memory
                        02AD   782 ;
                        02AD   783 ; SIDE EFFECTS:
                        02AD   784 ;
                        02AD   785 ;       none
                        02AD   786 ;--
                        02AD   787
                        02AD   788 CHECK_SPC:
                        02AD   789          ;
                        02AD   790          ; Registers R7 and R8 are saved at this level and may be used by
                        02AD   791          ; the action routines without being saved.  Action routines are JSB'ed
                        02AD   792          ; to with R7 containing the address of LOCAL_SPACE on the stack.
                        02AD   793          ;
     7E   57   7D       02AD   794          MOVQ    R7,-(SP)                ; Save registers
     57   0A   D0       02B0   795          MOVL    #SPECIAL_LEN,R7         ; Get number of table entries
58   FDA5 CF   DE       02B3   796          MOVAL   SPECIAL,R8              ; Get address of table
```

```
                                02B8    797
                                02B8    798 10$:    CMPW    R1,(R8)+                ; Does entry match item?
                        88  51  B1  02B8    799         BEQL    20$                 ; Yes, go handle it
                            08  13  02BB    800         ADDL    #4,R8               ; Skip handler address
                        58  04  C0  02BD    801         SOBGTR  R7,10$              ; Scan rest of table
                          F5 57  F5  02C0    802         BRB     30$                 ; Nothing to do, exit
                            09  11  02C3    803
                                02C5    804 20$:    MOVAL   LOCAL_SPACE(FP),R7      ; Load local address for action routine
                    57    F8 AD  DE  02C5    805         MOVZBL  S^#SS$_NORMAL,R0    ; Assume success
                        50  01  9A  02C9    806         JSB     @(R8)+              ; Call action routine
                            98  16  02CC    807
                                02CE    808 30$:    MOVQ    (SP)+,R7                ; Restore registers
                        57  8E  7D  02CE    809         RSB
                            05  02D1    810 ;+
                                02D2    811 ; Data handling routines
                                02D2    812 ;-
                                02D2    813
                                02D2    814 ;
                                02D2    815 ;  The PID must be returned as an EPID.
                                02D2    816 ;  The EPID field of the LKB is valid only on a master copy lock block.
                                02D2    817 ;
                                02D2    818 ; Inputs:       R4 -> LKB$L_EPID in LKB
                                02D2    819 ;               R7 -> Output longword buffer if needed for return
                                02D2    820 ;               R9 =  Address of LKB
                                02D2    821 ;
                                02D2    822
                                02D2    823 SPC_PID:
                            04  E0  02D2    824         BBS     #LKB$V_MSTCPY,-     ; Br if master copy, R4 is pointing to
                      10 2A A9              825                 LKB$W_STATUS(R9),90$ ;  a valid EPID
                    50    0C A9  D0  02D7    826         MOVL    LKB$L_PID(R9),R0    ; Else, get the IPID
                00000000'EF     16  02DB    827         JSB     EXE$IPID_TO_EPID    ; Convert to EPID
                        67  50  D0  02E1    828         MOVL    R0,(R7)             ; Store the EPID
                        54  57  D0  02E4    829         MOVL    R7,R4               ; Change the item address
                        50  01  9A  02E7    830 90$:    MOVZBL  S^#SS$_NORMAL,R0    ; Return success
                            05  02EA    831         RSB
                                02EB    832
                                02EB    833 ;
                                02EB    834 ; The lock state is a composite of several fields
                                02EB    835 ;
                                02EB    836
                                02EB    837 SPC_STATE:
                                02EB    838         ASSUME  LKB$B_GRMODE EQ LKB$B_RQMODE+1
                                02EB    839         ASSUME  LKB$B_STATE EQ LKB$B_GRMODE+1
                        67  84  3C  02EB    840         MOVZWL  (R4)+,(R7)          ; Copy modes
                    02 A7   64  90  02EE    841         MOVB    (R4),2(R7)          ; ..and state
                            05  18  02F2    842         BGEQ    30$                 ; Br if state is okay
                02 A7    FF 8F  90  02F4    843         MOVB    #LKI$C_WAITING,2(R7) ; Else, map waiting states to same code
                        54  57  D0  02F9    844 30$:    MOVL    R7,R4               ; Change the item address
                            05  02FC    845         RSB
                                02FD    846
                                02FD    847 ;
                                02FD    848 ; The lock's parent lock ID must be extracted from another LKB
                                02FD    849 ;
                                02FD    850
                                02FD    851 SPC_PARENT:
                        67  D4  02FD    852         CLRL    (R7)                ; Assume no PARENT LKB
                    54  64  D0  02FF    853         MOVL    (R4),R4             ; Get address of PARENT LKB
```

```
              04  13 0302             BEQL    10$                      ; Br if none
      67  30  A4  D0 0304             MOVL    LKB$L_LKID(R4),(R7)      ; Get LOCKID of owner process
          54  57  D0 0308     10$:    MOVL    R7,R4                    ; Change the item address
              05     030B             RSB
                     030C
                     030C     ;
                     030C     ; The CSID of master
                     030C     ;
                     030C
                     030C     SPC_SYSTEM:
              64  D5 030C             TSTL    (R4)                     ; Is CSID zero?
              10  12 030E             BNEQ    30$                      ; Br if not, CSID is okay
 50   00000000'EF D0 0310             MOVL    L^CLU$GL_CLUB,R0         ; Get address of cluster block
              04  13 0317             BEQL    20$                      ; Br if no cluster
      54  60 A0   9E 0319             MOVAB   CLUB$L_LOCAL_CSID(R0),R4 ; Set new item address
          50  01  9A 031D     20$:    MOVZBL  S^#SS$_NORMAL,R0         ; Return success
              05     0320     30$:    RSB
                     0321
                     0321     ;
                     0321     ; The lock's resource name space is a composite
                     0321     ;
                     0321
                     0321     SPC_NAMSPACE:
                     0321             ASSUME  RSB$B_RMOD EQ RSB$W_GROUP+2
          18  00  EF 0321             EXTZV   #0,#8*16,-               ; Get the group field and access mode
          67  64     0324                     (R4),(R7)                ;  3 bytes.
              64  B5 0326             TSTW    (R4)                     ; Is this group 0? (ie SYSTEM resource)
              04  12 0328             BNEQ    10$                      ; Br if not, not a system resource
   00 67  1F      E2 032A             BBSS    #LKI$V_SYSNAM,(R7),10$   ; Set the SYSTEM wide indicator
          54  57  D0 032E     10$:    MOVL    R7,R4                    ; Change the item address
              05     0331             RSB
                     0332
                     0332     ;
                     0332     ; The lock's lock count is the sum of all locks granted on the resource.
                     0332     ;
                     0332
                     0332     SPC_LCKCOUNT:
              67  D4 0332             CLRL    (R7)                     ; No locks granted yet!
          58  54  D0 0334             MOVL    R4,R8                    ; Copy listhead address
          58  64  D1 0337     10$:    CMPL    (R4),R8                  ; Back at listhead again?
              07  13 033A             BEQL    20$                      ; Br if yes
              67  D6 033C             INCL    (R7)                     ; Else, tally one more lock
          54  64  D0 033E             MOVL    (R4),R4                  ; move down list
              F4  11 0341             BRB     10$                      ; Look for more
          54  57  D0 0343     20$:    MOVL    R7,R4                    ; Change item address
              05     0346             RSB
                     0347
                     0347     ;
                     0347     ; The remote lock id
                     0347     ;
                     0347
                     0347     SPC_REMLKID:
              51  DD 0347             PUSHL   R1                       ; Save R1
      51  50  A9  D0 0349             MOVL    LKB$L_RSB(R9),R1         ; Get RSB address
      67  38  A1  D0 034D             MOVL    RSB$L_CSID(R1),(R7)      ; Is the REMLKID valid?
              03  13 0351             BEQL    10$                      ; Br if not, information is still local
          67  64  D0 0353             MOVL    (R4),(R7)                ; Else, return real REMLKID
```

```
                                                E 3
                54    57    D0   0356   911 10$:     MOVL    R7,R4                   ; Return item address
                      51 8ED0   0359   912           POPL    R1                      ; Restore R1
                         05     035C   913           RSB                             ; Return to caller
                                035D   914
                                035D   915 ;
                                035D   916 ; The list of all locks being blocked by this lock.
                                035D   917 ;
                                035D   918
                                035D   919 SPC_BLOCKEDBY:
                      06        BB   035D   920           PUSHR   #^M<R1,R2>              ; Save registers
                    0367        30   035F   921           BSBW    LKI_ALLOCATE            ; Allocate a system buffer
                   2A 50        E9   0362   922           BLBC    R0,50$                  ; Br if resource failure
                   58   54      D0   0365   923           MOVL    R4,R8                   ; Copy RSB wait queue listhead address
                   54   52      D0   0368   924           MOVL    R2,R4                   ; Copy address of system buffer data
                        04     E0   036B   925           BBS     #LKB$V_MSTCPY,-         ; Br if this is the master copy,
                12 2A A9               036D   926                   LKB$W_STATUS(R9),10$    ;  information is local to this system
                53   50 A9    D0   0370   927           MOVL    LKB$L_RSB(R9),R3        ; Get RSB address
                53   38 A3    D0   0374   928           MOVL    RSB$L_CSID(R3),R3       ; Is this a process copy?
                        08     13   0378   929           BEQL    10$                     ; Br if not, information is still local
                                037A   930           ;
                                037A   931           ; Lock information is on MASTER system
                                037A   932           ;
            00000000'GF         16   037A   933           JSB     G^LKI$SND_BLKBY         ; Send request for all locks BLOCKEDBY
                                0380   934                                           ;  this lock
                        03      11   0380   935           BRB     30$                     ; Return with status
                                0382   936           ;
                                0382   937           ; Lock information is LOCAL to this system
                                0382   938           ;
                      0288      30   0382   939 10$:     BSBW    LKI$SEARCH_BLOCKEDBY    ; Find all locks BLOCKEDBY this lock
                   53   18      B0   0385   940 30$:     MOVW    #LKI$C_LENGTH,R3        ; Return size of item
            53      53   10      78   0388   941           ASHL    #16,R3,R3               ; Move to high word
                   53   6A      B0   038C   942           MOVW    (R10),R3                ; Get size of returned buffer
                        06      BA   038F   943 50$:     POPR    #^M<R1,R2>              ; Restore registers
                                05   0391   944           RSB
                                0392   945
                                0392   946 ;
                                0392   947 ; The list of all locks blocking this lock.
                                0392   948 ;
                                0392   949
                                0392   950 SPC_BLOCKING:
                      06        BB   0392   951           PUSHR   #^M<R1,R2>              ; Save registers
                    0332        30   0394   952           BSBW    LKI_ALLOCATE            ; Allocate a system buffer
                   2A 50        E9   0397   953           BLBC    R0,50$                  ; Br if resource failure
                   58   54      D0   039A   954           MOVL    R4,R8                   ; Copy RSB wait queue listhead address
                   54   52      D0   039D   955           MOVL    R2,R4                   ; Copy address of system buffer data
                        04     E0   03A0   956           BBS     #LKB$V_MSTCPY,-         ; Br if this is the master copy,
                12 2A A9               03A2   957                   LKB$W_STATUS(R9),10$    ;  information is local to this system
                53   50 A9    D0   03A5   958           MOVL    LKB$L_RSB(R9),R3        ; Get RSB address
                53   38 A3    D0   03A9   959           MOVL    RSB$L_CSID(R3),R3       ; Is this a process copy?
                        08     13   03AD   960           BEQL    10$                     ; Br if not, information is still local
                                03AF   961           ;
                                03AF   962           ; Lock information is on MASTER system
                                03AF   963           ;
            00000000'GF         16   03AF   964           JSB     G^LKI$SND_BLKING        ; Send request for all locks BLOCKING
                                03B5   965                                           ;  this lock
                        03      11   03B5   966           BRB     30$                     ; Return with status
                                03B7   967           ;
```

```
                     03B7   968          ; Lock information is LOCAL to this system
                     03B7   969          ;
          01B2   30  03B7   970 10$:      BSBW    LKI$SEARCH_BLOCKING     ; Find all locks BLOCKING this lock
       53   18   B0  03BA   971 30$:      MOVW    #LKI$C_LENGTH,R3        ; Return size of item
 53    53   10   78  03BD   972           ASHL    #16,R3,R3               ; Move to high word
       53   6A   B0  03C1   973           MOVW    (R10),R3                ; Get size of returned buffer
            06   BA  03C4   974 50$:      POPR    #^M<R1,R2>              ; Restore registers
                 05  03C6   975           RSB
                     03C7   976
                     03C7   977 ;
                     03C7   978 ; The list of all locks associated with the resource.
                     03C7   979 ;
                     03C7   980
                     03C7   981 SPC_LOCKS:
            06   BB  03C7   982           PUSHR   #^M<R1,R2>              ; Save registers, R3 & R4 are outputs
          02FD   30  03C9   983           BSBW    LKI_ALLOCATE            ; Allocate a system buffer
       52 50   E9  03CC   984           BLBC    R0,80$                  ; Br if failure
    58    54   D0  03CF   985           MOVL    R4,R8                   ; Copy listhead address
    54    52   D0  03D2   986           MOVL    R2,R4                   ; Set address of return buffer
            04   E0  03D5   987           BBS     #LKB$V_MSTCPY,-         ; Br if this is the master copy,
 12 2A A9           03D7   988                   LKB$W_STATUS(R9),10$    ;  information is local to this system
 53    50 A9   D0  03DA   989           MOVL    LKB$L_RSB(R9),R3        ; Get RSB address
 53    38 A3   D0  03DE   990           MOVL    RSB$L_CSID(R3),R3       ; Is this a process copy?
            08   13  03E2   991           BEQL    10$                     ; Br if not, information is still local
                     03E4   992           ;
                     03E4   993           ; Lock information is on MASTER system
                     03E4   994           ;
00000000'GF  16   03E4   995           JSB     G^LKI$SND_LOCKS         ; Send request for all locks associated
                     03EA   996                                           ;  with this lock
            2B   11  03EA   997           BRB     70$                     ; Return with status
                     03EC   998           ;
                     03EC   999           ; Lock information is LOCAL to this system
                     03EC  1000           ;
    51   56   D0  03EC  1001 10$:      MOVL    R6,R1                   ; Get size of buffer
                     03EF  1002           ASSUME  RSB$L_CVTQFL EQ RSB$L_GRQFL+8
                     03EF  1003           ASSUME  RSB$L_WTQFL EQ RSB$L_CVTQFL+8
    53   03   9A  03EF  1004           MOVZBL  #3,R3                   ; Initialize number of queues to search
    57   58   D0  03F2  1005 30$:      MOVL    R8,R7                   ; Copy listhead address, again
    58   67   D1  03F5  1006 50$:      CMPL    (R7),R8                 ; Back at listhead again?
         14   13  03F8  1007           BEQL    60$                     ; Br if yes
    51   18   C2  03FA  1008           SUBL    #LKI$C_LENGTH,R1        ; Any room left in buffer?
         25   19  03FD  1009           BLSS    90$                     ; Br if not
    57   67   D0  03FF  1010           MOVL    (R7),R7                 ; Else, move down list
 57   C8 A7   9E  0402  1011           MOVAB   -LKB$L_SQFL(R7),R7      ; Point to start of LKB
         23   10  0406  1012           BSBB    LOCK_INFO               ; Get the lock information
 57   38 A7   9E  0408  1013           MOVAB   LKB$L_SQFL(R7),R7       ; Point back to state queue
         E7   11  040C  1014           BRB     50$                     ; Look for more
                  040E  1015 60$:      ASSUME  RSB$L_CVTQFL EQ RSB$L_GRQFL+8
                  040E  1016           ASSUME  RSB$L_WTQFL EQ RSB$L_CVTQFL+8
    58   08   C0  040E  1017           ADDL    #8,R8                   ; Skip to next queue
      DE 53   F5  0411  1018           SOBGTR  R3,30$                  ; Loop if more queues to search
    50   01   9A  0414  1019           MOVZBL  S^#SS$_NORMAL,R0        ; Return success
    53   18   B0  0417  1020 70$:      MOVW    #LKI$C_LENGTH,R3        ; Return size of item
 53 53   10   78  041A  1021           ASHL    #16,R3,R3               ; Move to high word
    53   6A   B0  041E  1022           MOVW    (R10),R3                ; Get size of returned buffer
         06   BA  0421  1023 80$:      POPR    #^M<R1,R2>              ; Restore registers
              05  0423  1024           RSB                             ; Return to caller
```

```
                     0424  1025
50   0601 BF   3C    0424  1026 90$:    MOVZWL  #SS$_BUFFEROVF,R0       ; Return partial success
          F6   11    0429  1027          BRB     80$                     ; Exit
                     042B  1028
                     042B  1029 ;+
                     042B  1030 ; Return Lock Information
                     042B  1031 ;
                     042B  1032 ;       This routine will return the following lock information:
                     042B  1033 ;
                     042B  1034 ;               LKI$_LOCKID     - the lock's lock id
                     042B  1035 ;               LKI$_PID        - the lock's PID
                     042B  1036 ;               LKI$_SYSTEM     - the resource's system id
                     042B  1037 ;               LKI$_STATE      - the locks current state
                     042B  1038 ;               LKI$_REMLKID    - the remote lock id (Process copy LOCKID)
                     042B  1039 ;               LKI$_REMSYSTEM  - the remote system id (Process copy CSID)
                     042B  1040 ;
                     042B  1041 ; Inputs:
                     042B  1042 ;       R2 = Output buffer address
                     042B  1043 ;       R7 = LKB address
                     042B  1044 ;       R10 = Address of beginning of system buffer
                     042B  1045 ;
                     042B  1046 ; Outputs:
                     042B  1047 ;       None
                     042B  1048 ;
                     042B  1049 ; Side Effects:
                     042B  1050 ;       R0 is destroyed
                     042B  1051 ;       (R10) is increased by lock return size
                     042B  1052 ;-
                     042B  1053 LOCK_INFO:
     6A   18   A0    042B  1054          ADDW    #LKI$C_LENGTH,(R10)     ; Tally return size
     82   30   A7    042E  1055          MOVL    LKB$L_LKID(R7),(R2)+    ; Return the LOCKID (MASTER LOCKID)
                     0432  1056          ;
                     0432  1057          ;  The EPID in the LKB is valid only for a master lock block.
                     0432  1058          ;
     50   14   A7    0432  1059          MOVL    LKB$L_EPID(R7),R0       ; Get the EPID
               04    0436  1060          BBS     #LKB$V_MSTCPY,-         ; Br if master copy lock
        0A   2A   A7 0438  1061                  LKB$W_STATUS(R7),10$    ; ...EPID is valid
     50   0C   A7    043B  1062          MOVL    LKB$L_PID(R7),R0        ; Get the IPID
00000000'EF   16    043F  1063          JSB     L^EXE$IPID_TO_EPID      ; Convert to EPID
     82   50   D0    0445  1064 10$:     MOVL    R0,(R2)+                ; Return the EPID
     50   50   A7    0448  1065          MOVL    LKB$L_RSB(R7),R0        ; Get RSB address
     82   38   A0    044C  1066          MOVL    RSB$L_CSID(R0),(R2)+    ; Return the SYSTEM ID (MASTER CSID)
               0E    0450  1067          BNEQ    30$                     ; Br if okay
50   00000000'EF   D0    0452  1068          MOVL    L^CLU$GL_CLUB,R0        ; Else, get address of cluster block
               05    0459  1069          BEQL    30$                     ; Br if no cluster
   FC A2   60   A0    D0    045B  1070          MOVL    CLUB$L_LOCAL_CSID(R0),-4(R2) ; Return real CSID
                     0460  1071 30$:     ASSUME  LKB$B_GRMODE+1 EQ LKB$B_RQMODE
     82   34   A7    B0    0460  1072          MOVW    LKB$B_RQMODE(R7),(R2)+  ; Copy modes
     82   36   A7    9B    0464  1073          MOVZBW  LKB$B_STATE(R7),(R2)+   ; Copy current state, zero byte
               05    18    0468  1074          BGEQ    40$                     ; Br if state is okay
   FE A2   FF   8F    90    046A  1075          MOVB    #LKI$C_WAITING,-2(R2)   ; Else, map waiting states to same code
                     046F  1076 40$:     ;
                     046F  1077          ;  The remote CSID and REMLKID are only valid in a master copy
                     046F  1078          ;  lock block.
                     046F  1079          ;
     82   54   A7    D0    046F  1080          MOVL    LKB$L_REMLKID(R7),(R2)+ ; Copy the REMLKID (PROCESS COPY LKID)
     82   58   A7    D0    0473  1081          MOVL    LKB$L_CSID(R7),(R2)+    ; Get the remote CSID (PROCESS_CPY CSID)
```

```
                                                     H 3
                 04   E0  0477  1082          BBS     #LKB$V_MSTCPY,-          ; Br if master copy
               16 2A A7      0479  1083                  LKB$W_STATUS(R7),90$     ; ...CSID, REMLKID are valid
            FB A2   50 A7    D0  047C  1084          MOVL    LKB$L_LKID(R7),-8(R2)    ; Else, return the LOCKID as REMLKID
         50   00000000'EF    D0  0481  1085          MOVL    L^CLU$GL_CLUB,R0         ; Get the CLUB
                 04   13    0488  1086          BEQL    70$                      ; Br if none, return zero CSID
            50   60 A0      D0  048A  1087          MOVL    CLUB$L_LOCAL_CSID(R0),R0; Else, get real CSID
            FC A2   50      D0  048E  1088 70$:     MOVL    R0,-4(R2)                ; Return real CSID
                            05  0492  1089 90$:     RSB
                                0493  1090
```

```
                     1092          .SBTTL  GETLKB - Get specified Lock Block
                     1093 ;++
                     1094 ;
                     1095 ; FUNCTIONAL DESCRIPTION:
                     1096 ;
                     1097 ;       Routine to convert a LKID and check privileges. If a valid LKID is
                     1098 ;       specified, the standard conversion routine VERIFYLOCKID is simply
                     1099 ;       called.  If, however, a LKID that implies a "wildcard" LKID (-1 or 0)
                     1100 ;       is specified, then the next active lock is chosen as the LKID to pass
                     1101 ;       to VERIFYLOCKID which then checks the requestor's privilege to obtain
                     1102 ;       information about the lock and returns the lock's LKB address.
                     1103 ;
                     1104 ; CALLING SEQUENCE:
                     1105 ;
                     1106 ;       JSB/BSB
                     1107 ;
                     1108 ; INPUTS:
                     1109 ;
                     1110 ;       R4              current process PCB address
                     1111 ;       LKID(AP)        address of specified LKID
                     1112 ;
                     1113 ; IMPLICIT INPUTS:
                     1114 ;
                     1115 ;       IPL <= IPL$_ASTDEL
                     1116 ;
                     1117 ; OUTPUTS:
                     1118 ;
                     1119 ;       R0      success/failure of operation
                     1120 ;       R4      current process PCB address
                     1121 ;       R9      specified lock's LKB address
                     1122 ;
                     1123 ; COMPLETION CODES:
                     1124 ;
                     1125 ;       SS$_NORMAL      Normal successful completion
                     1126 ;       SS$_ACCVIO      Access violation on attempt to access lock id
                     1127 ;       SS$_NOMORELOCK  No more locks available (on "wildcard" operations)
                     1128 ;
                     1129 ; SIDE EFFECTS:
                     1130 ;
                     1131 ;       R5 and R6 are destroyed.
                     1132 ;--
                     1133
                     1134 GETLKB:
          55    D4   0493  1135          CLRL    R5                      ; Assume not "wildcard" LKID
     56   08 AC    D0   0495  1136          MOVL    LKID(AP),R6             ; Get LKID address
          42    13   0499  1137          BEQL    60$                     ; Br if none
                     049B  1138          IFNOWRT #4,(R6),50$             ; Check access to LKID
          51    66   D0   04A1  1139          MOVL    (R6),R1                 ; Get LKID
          23    14   04A4  1140          BGTR    20$                     ; Br if standard LKID
                     04A6  1141          ;
                     04A6  1142          ; "Wildcard" type LKID specified
                     04A6  1143          ;
          55    51   32   04A6  1144          CVTWL   R1,R5                   ; Get LKIX (LocK IndeX) from LKID
          02    14   04A9  1145          BGTR    10$                     ; If gtr, valid LKIX
          55    D4   04AB  1146          CLRL    R5                      ; Else, start with index = 1
          55    B6   04AD  1147 10$:     INCW    R5                      ; Increment LKIX
00000000'EF     55    B1   04AF  1148          CMPW    R5,LCK$GL_MAXID         ; Is LKIX in valid range?
```

```
                     25   1A  04B6  1149          BGTRU   60$                     ; Br if not - no more locks
          50 00000000'FF45   D0  04B8  1150          MOVL    @LCK$GL_IDTBL[R5],R0    ; Get LKB address
                     EB   18  04C0  1151          BGEQ    10$                     ; Br if unused slot
                  51 30 A0   D0  04C2  1152          MOVL    LKB$L_LKID(R0),R1       ; Get LKID from LKB
                     66 51   D0  04C6  1153          MOVL    R1,(R6)                 ; Store LKID in argument list
                             04C9  1154          ;
                             04C9  1155          ; Get LKB and check privileges
                             04C9  1156          ;
                        0018 30  04C9  1157  20$:   BSBW    VERIFYLOCKID            ; Get LKB address and check privileges
                          55 B5  04CC  1158          TSTW    R5                      ; "wildcard" type LKID specified?
                          07 13  04CE  1159          BEQL    40$                     ; Br if not
                       DA 50 E9  04D0  1160          BLBC    R0,10$                  ; Br if error, return only "good" ones
                  02 A6    01 AE  04D3  1161          MNEGW   #1,2(R6)                ; Else, set continuation context
                             05  04D7  1162  40$:   RSB                             ; Return to caller
                             04D8  1163          ;
                          50 0C 3C  04D8  1164  50$:   MOVZWL  #SS$_ACCVIO,R0          ; Set access violation
                             FA 11  04DB  1165          BRB     40$                     ;
                50 0A08 8F    3C  04DD  1166  60$:   MOVZWL  #SS$_NOMORELOCK,R0      ; Set no more processes
                             F3 11  04E2  1167          BRB     40$                     ;
                             04E4  1168
```

```
                                                                .SBTTL  VERIFYLOCKID - Verify lock id

                                                        ;++
                                                        ; FUNCTIONAL DESCRIPTION:
                                                        ;
                                                        ;       This routine verifies a lock id for correct process ownership
                                                        ;       and access mode and then converts it into a LKB address.
                                                        ;
                                                        ;       LKB is not locked after leaving this routine, therefore we
                                                        ;       must re-verify the LKB everytime we attempt to use it.
                                                        ;
                                                        ; CALLING SEQUENCE:
                                                        ;
                                                        ;       JSB/BSB
                                                        ;
                                                        ;       Note:  IPL is raised to IPL$_SYNCH to prevent the owner of
                                                        ;       the lock from releasing the LKB/RSB in the middle of verifying
                                                        ;       its lock id.
                                                        ;
                                                        ; INPUTS:
                                                        ;
                                                        ;       R1      Lock id
                                                        ;       R4      Address of PCB
                                                        ;       R5      Zero if not a wildcard search operation
                                                        ;
                                                        ; OUTPUTS:
                                                        ;
                                                        ;       R0      Completion code
                                                        ;       R9      Address of LKB
                                                        ;
                                                        ; COMPLETION CODES:
                                                        ;
                                                        ;       SS$_NORMAL      Lock id was valid and converted to LKB address
                                                        ;       SS$_IVLOCKID    Invalid lock id
                                                        ;       SS$_IVMODE      Access mode violation on attempt to access lock
                                                        ;       SS$_NOSYSLCK    No SYSLCK privilege to access system lock
                                                        ;       SS$_NOWORLD     No WORLD privilege to access lock
                                                        ;
                                                        ; SIDE EFFECTS:
                                                        ;
                                                        ;       R0 and R1 are destroyed
                                                        ;--
                                                        ;
                                                                ASSUME  LKB$V_MODE  EQ  0
                                                                ASSUME  LKB$S_MODE  EQ  2
                                                        ;
                                                        VERIFYLOCKID:
                                                                DSBINT  #IPL$_SYNCH             ; Raise IPL to sync access to LKBs
                                                                MOVZWL  R1,R9                   ; Put lockid index in R9
                                00000000'EF   59    D1  04ED  CMPL    R9,LCK$GL_MAXID       ; Is the lock id too big?
                                              5A    1A  04F4  BGTRU   40$                   ; Yes
                            59  00000000'FF49    D0  04F6  MOVL    @LCK$GL_IDTBL[R9],R9  ; Get LKB address
                                              50    18  04FE  BGEQ    40$                   ; Unallocated id
                                    30 A9     51    D1  0500  CMPL    R1,LKB$L_LKID(R9)     ; Check sequence number
                                              4A    12  0504  BNEQ    40$                   ; Not valid
                                    50   50 A9     D0  0506  MOVL    LKB$L_RSB(R9),R0      ; Get RSB address
                                           4C A0    B5  050A  TSTW    RSB$W_GROUP(R0)       ; Is this a system resource?
```

```
                        17   13  050D  1227          BEQL    10$                     ; Br if yes
          51  00000000'GF   D0  050F  1228          MOVL    G^SCH$GL_CURPCB,R1      ; Else, get our PCB address
                00BE C1   B1  0516  1229          CMPW    PCB$W_GRP(R1),-         ; Do we have group access to LKB?
                   4C A0      051A  1230                  RSB$W_GROUP(R0)         ; ..no privilege needed
                        1A   13  051C  1231          BEQL    20$                     ; Br if our group - always allowed
                                 051E  1232          IFNPRIV WORLD,70$               ; Br if NO privilege to access lock
                        12   11  0524  1233          BRB     20$                     ; Else, success
                        50   DC  0526  1234 10$:      MOVPSL  R0                      ; Get current PSL
                        16   EF  0528  1235          EXTZV   #PSL$V_PRVMOD,-         ; Extract previous mode field
              50    50    02        052A  1236                  #PSL$S_PRVMOD,R0,R0
                                 052D  1237          ASSUME  PSL$C_KERNEL EQ 0
                                 052D  1238          ASSUME  PSL$C_EXEC EQ 1
                   50   01   91  052D  1239          CMPB    #PSL$C_EXEC,R0          ; Does the user have the right access
                                 0530  1240                                          ;  mode to access the LKB?
                        06   1E  0530  1241          BGEQU   20$                     ; Br if yes
                                 0532  1242          IFNPRIV SYSLCK,60$              ; Br if NO privilege to look at lock
                        50   DC  0538  1243 20$:      MOVPSL  R0                      ; Get current PSL
                        16   EF  053A  1244          EXTZV   #PSL$V_PRVMOD,-         ; Extract previous mode field
              50    50    02        053C  1245                  #PSL$S_PRVMOD,R0,R0
                51    50  A9   D0  053F  1246          MOVL    LKB$L_RSB(R9),R1        ; Get RSB address
                4E  A1    50   91  0543  1247          CMPB    R0,RSB$B_RMOD(R1)       ; Caller have privilege to access lock?
                        0E   1A  0547  1248          BGTRU   50$                     ; Br if No
                   50   01   9A  0549  1249          MOVZBL  S^#SS$_NORMAL,R0        ; Else, Yes - return success
                                 054C  1250 30$:      ENBINT                          ; Restore IPL
                        05       054F  1251          RSB
                                 0550  1252
              50  2124  8F   3C  0550  1253 40$:      MOVZWL  #SS$_IVLOCKID,R0        ; Invalid lock id
                        F5   11  0555  1254          BRB     30$                     ; Leave
              50  0354  8F   3C  0557  1255 50$:      MOVZWL  #SS$_IVMODE,R0          ; Illegal access mode
                        EE   11  055C  1256          BRB     30$                     ; Leave
              50  28F4  8F   3C  055E  1257 60$:      MOVZWL  #SS$_NOSYSLCK,R0        ; No SYSLCK privilege to access lock
                        E7   11  0563  1258          BRB     30$                     ; Leave
              50  2884  8F   3C  0565  1259 70$:      MOVZWL  #SS$_NOWORLD,R0         ; No WORLD privilege to access lock
                        E0   11  056A  1260          BRB     30$                     ; Leave
                                 056C  1261
```

```
                        1263          .SBTTL  LKI$SEARCH_BLOCKING - Search for locks blocking the current lock
                        1264
                        1265 ;++
                        1266 ; FUNCTIONAL DESCRIPTION:
                        1267 ;
                        1268 ;       This routine searches for locks blocking the current lock.  A
                        1269 ;       blocking lock is one in which the maximized request mode is
                        1270 ;       incompatible with the requested mode (if the lock is on the
                        1271 ;       waiting or conversion queue) or the granted mode (if the lock
                        1272 ;       is on the granted queue).
                        1273 ;
                        1274 ;       For example, assume there is PR locks granted on a resource and
                        1275 ;       a second user issues an EX mode request on the resource. The first
                        1276 ;       lock is now BLOCKING the second lock and the first lock would be
                        1277 ;       returned in list of locks BLOCKING the second lock.
                        1278 ;
                        1279 ;       To find BLOCKING locks it is sufficient to check all locks
                        1280 ;       ahead of this lock on all queues (in th order, REQESTED,
                        1281 ;       CONVERSION and then GRANTED) to see if their requested or granted
                        1282 ;       modes are incompatible with this locks requested mode.
                        1283 ;
                        1284 ; CALLING SEQUENCE:
                        1285 ;
                        1286 ;       JSB/BSB
                        1287 ;
                        1288 ; INPUTS:
                        1289 ;
                        1290 ;       R2      address of system buffer for storing the lock information
                        1291 ;       R6      length of system buffer for storing the lock information
                        1292 ;       R8      address of wait queue in RSB
                        1293 ;       R9      LKB address
                        1294 ;
                        1295 ; IMPLICIT INPUTS:
                        1296 ;
                        1297 ;       IPL = IPL$_SYNCH
                        1298 ;
                        1299 ; OUTPUTS:
                        1300 ;
                        1301 ;       R0      always success!
                        1302 ;
                        1303 ; SIDE EFFECTS:
                        1304 ;
                        1305 ;       R7 is destroyed.
                        1306 ;--
                        1307
                        1308 LKI$SEARCH_BLOCKING::
0066 8F    BB           1309          PUSHR   #^M<R1,R2,R5,R6>        ; Save registers
                        1310          ;
                        1311          ; First run through all locks waiting ahead of this lock
                        1312          ; maximizing the requested modes and checking all locks
                        1313          ; incompatible with the current 'maxmode".  If this lock is
                        1314          ; on the wait queue then we do the wait queue first and
                        1315          ; the converison queue next.  If this lock is on the
                        1316          ; conversion queue then we do only the conversion queue.
                        1317          ; Later we'll do all the granted locks.
                        1318          ;
                        1319          ; If this lock is on the granted queue, we skip right to the
```

```
                     0570  1320          ; search of the granted queue locks.
                     0570  1321          ;
                     0570  1322          ASSUME  LKB$K_GRANTED  EQ  1
                     0570  1323          ASSUME  LKB$K_CONVERT  EQ  0
                     0570  1324          ASSUME  LKB$K_WAITING  EQ  -1
                     0570  1325          ASSUME  RSB$L_CVTQFL   EQ  RSB$L_GRQFL+8
                     0570  1326          ASSUME  RSB$L_WTQFL   EQ  RSB$L_CVTQFL+8
                     0570  1327
  55   34 A9   9A    0570  1328          MOVZBL  LKB$B_RQMODE(R9),R5      ; Get the current lock's requested mode
  57   59 D0         0574  1329          MOVL    R9,R7                   ; R7 will point to other LKB's
                     0577  1330                                          ; in front of the one pointed to by R9
    36 A9 95         0577  1331          TSTB    LKB$B_STATE(R9)         ; Which queue is lock on?
       63 14         057A  1332          BGTR    60$                     ; Br if granted queue
       03 19         057C  1333          BLSS    10$                     ; Br if waiting queue
                     057E  1334          ;
                     057E  1335          ;   Lock is on the conversion queue
                     057E  1336          ;
  58   08 C2         057E  1337          SUBL    #8,R8                   ; Point to conversion queue header
                     0581  1338
  57   3C A7   D0    0581  1339 10$:     MOVL    LKB$L_SQBL(R7),R7       ; Get previous lock on state queue
  58   57 D1         0585  1340          CMPL    R7,R8                   ; Reached head of queue yet?
       42 13         0588  1341          BEQL    50$                     ; Br if yes
  57   38 C2         058A  1342          SUBL    #LKB$L_SQFL,R7          ; Back up to point at start of LKB
  50   34 A7   9A    058D  1343          MOVZBL  LKB$B_RQMODE(R7),R0     ; R0 = requested mode
  51   55 D0         0591  1344          MOVL    R5,R1                   ; Save old maxmode
                     0594  1345          ;
                     0594  1346          ; Maximize lock modes (in R0 and R5) and see if this lock (R7) is
                     0594  1347          ; incompatible with (the previous) maxmode.  The maximization function
                     0594  1348          ; is a simple arithmetic maximum except if the two modes are CW and PR.
                     0594  1349          ; In that case the maximum of CW and PR is PW.  PW is incompatible
                     0594  1350          ; with everything either CW or PR is incompatible with.
                     0594  1351          ;
  55   50 91         0594  1352          CMPB    R0,R5                   ; Current mode greater than maxmode?
       20 13         0597  1353          BEQL    35$                     ; Br if No, they're equal
       0C 1A         0599  1354          BGTRU   20$                     ; Br if Yes, compute new maxmode
  02   50 91         059B  1355          CMPB    R0,#LCK$K_CWMODE        ; Br if No, is current mode CW?
       19 12         059E  1356          BNEQ    35$                     ; Br if No, maxmode = R2
  03   55 91         05A0  1357          CMPB    R5,#LCK$K_PRMODE        ; Br if Yes, is maxmode PR?
       14 12         05A3  1358          BNEQ    35$                     ; Br if No, maxmode = R2
       0A 11         05A5  1359          BRB     25$                     ; Br if Yes, new maxmode is PW
  02   55 91         05A7  1360 20$:     CMPB    R5,#LCK$K_CWMODE        ; Is maxmode CW?
       0A 12         05AA  1361          BNEQ    30$                     ; Br if No, maxmode = R0
  03   50 91         05AC  1362          CMPB    R0,#LCK$K_PRMODE        ; Br if Yes, is current mode PR?
       05 12         05AF  1363          BNEQ    30$                     ; Br if No, maxmode = R0
  55   04 90         05B1  1364 25$:     MOVB    #LCK$K_PWMODE,R5        ; Have CW and PR; maxmode = PW
       03 11         05B4  1365          BRB     35$
  55   50 90         05B6  1366 30$:     MOVB    R0,R5                   ; Maxmode = R0
                     05B9  1367
00000000'EF41  50 E0 05B9  1368 35$:     BBS     R0,-                    ; Branch if compatible with
       BF            05C1  1369                  L^LCK$COMPAT_TBL[R1],10$;  saved maxmode
                     05C2  1370          ;
                     05C2  1371          ; Have a lock incompatible with maxmode, return the lock info.
                     05C2  1372          ;
  56   18 C2         05C2  1373          SUBL    #LKI$C_LENGTH,R6        ; Any room left in buffer?
       3E 19         05C5  1374          BLSS    90$                     ; Br if not, leave now
    FE61 30          05C7  1375          BSBW    LOCK_INFO               ; Return the lock information
       B5 11         05CA  1376 40$:     BRB     10$                     ; Get next lock in RSB (outer loop)
```

```
                                05CC  1377
                                05CC  1378 50$:     ;
                                05CC  1379          ; Reached the queue header.  Back up R8 to point to the previous
                                05CC  1380          ; queue header in the RSB.  If R8 is pointing to the granted
                                05CC  1381          ; queue, then we are done with this loop and we continue with
                                05CC  1382          ; the granted queue.  Otherwise, we repeat this loop for the
                                05CC  1383          ; conversion queue.
                                05CC  1384          ;
          58    08    C2      05CC  1385          SUBL    #8,R8                   ; Back up R8 one queue header
      57  C8    A8    9E      05CF  1386          MOVAB   -LKB$L_SQFL(R8),R7      ; Prepare to process that queue
                10    C1      05D3  1387          ADDL3   #RSB$L_GRQFL,-          ; Get address of granted queue
      50  50    A9            05D5  1388                  LKB$L_RSB(R9),R0
          50    58    D1      05D8  1389          CMPL    R8,R0                   ; Have we reached the granted queue?
                ED    12      05DB  1390          BNEQ    40$                     ; Br if Not, repeat for conversion queue
                              05DD  1391          ;
                              05DD  1392          ; Now repeat a similar procedure for all locks on the granted
                              05DD  1393          ; queue whose granted mode is incompatible with the maxmode
                              05DD  1394          ; in R5.
                              05DD  1395          ;
                      03      11  05DD  1396          BRB     70$
                              05DF  1397
                              05DF  1398 60$:     ;
                              05DF  1399          ;  Lock is initially on the granted queue.
                              05DF  1400          ;
          58    10    C2      05DF  1401          SUBL    #16,R8                  ; Point to granted queue header
                              05E2  1402
      57  3C    A7    D0      05E2  1403 70$:     MOVL    LKB$L_SQBL(R7),R7       ; Get next lock in granted queue
          58    57    D1      05E6  1404          CMPL    R7,R8                   ; Reached end of queue?
                1A    13      05E9  1405          BEQL    90$                     ; Br if Yes, all done
          57    38    C2      05EB  1406          SUBL    #LKB$L_SQFL,R7          ; Back up to point at start of LKB
      50  35    A7    9A      05EE  1407          MOVZBL  LKB$B_GRMODE(R7),R0     ; Get granted mode
E7 00000000'EF45    50    E0  05F2  1408          BBS     R0,L^[CK$COMPAT_TBL[R5],70$ ; Branch if compatible
                              05FB  1409          ;
                              05FB  1410          ; Have an incompatible lock on the granted queue, return lock info.
                              05FB  1411          ;
              56    18    C2  05FB  1412          SUBL    #LKI$C_LENGTH,R6        ; Any room left in buffer?
                    05    19  05FE  1413          BLSS    90$                     ; Br if not, leave now
                FE28      30  0600  1414          BSBW    LOCK_INFO               ; Return lock info
                  DD      11  0603  1415          BRB     70$                     ; Look for more
                              0605  1416
              50    01    9A  0605  1417 90$:     MOVZBL  #1,R0                   ; Success indicator
              0066  BF    BA  0608  1418          POPR    #^M<R1,R2,R5,R6>        ; Restore registers
                        05    060C  1419          RSB
                              060D  1420
```

```
                                .SBTTL  LKI$SEARCH_BLOCKEDBY - Search for locks blockedby the current lock

                        ;++
                        ; FUNCTIONAL DESCRIPTION:
                        ;
                        ;       This routine searches for locks blocked by the current lock.
                        ;       A blocked lock is one which is either blocked by the current
                        ;       lock or is blocked by any other lock blocked by the current
                        ;       lock. We must start with the current lock on whatever queue
                        ;       it may currently be on and then maximize the requested for
                        ;       locks on the converting or waiting queues. All locks are checked
                        ;       to see if the maximized request mode is incompatible with the
                        ;       requested mode (if the locks is not on the granted queue).
                        ;
                        ;       For example, assume there is an EX lock granted on a resource and
                        ;       a two other users have issued PR requests on the resource. Now
                        ;       if we wish to find all locks BLOCKEDBY the first lock, then the
                        ;       list consists of the two locks waiting for the resource in PR
                        ;       mode.
                        ;
                        ;       To find BLOCKING locks it is sufficient to check all locks
                        ;       behing the current lock on all queues (in the order, GRANTED
                        ;       CONVERTING and then WAITING) to see if their requested mode
                        ;       is incompatible with the current lock's requested (or granted)
                        ;       mode. Once, we have found one blocked lock, then that lock and all
                        ;       locks following are also blocked.
                        ;
                        ; CALLING SEQUENCE:
                        ;
                        ;       JSB/BSB
                        ;
                        ; INPUTS:
                        ;
                        ;       R2      address of system buffer for storing the lock information
                        ;       R6      length of system buffer for storing the lock information
                        ;       R8      address of wait queue in RSB
                        ;       R9      LKB address
                        ;
                        ; IMPLICIT INPUTS:
                        ;
                        ;       IPL = IPL$_SYNCH
                        ;
                        ; OUTPUTS:
                        ;
                        ;       R0      always success!
                        ;
                        ; SIDE EFFECTS:
                        ;
                        ;       R7 is destroyed.
                        ;--
                        ;
                        LKI$SEARCH_BLOCKEDBY::
0066 8F    BB           PUSHR   #^M<R1,R2,R5,R6>        ; Save registers
                        ;
                        ; First run through all locks waiting behind this lock
                        ; maximizing the requested modes and checking all locks
                        ; incompatible with the current "maxmode".  If we find a
```

```
                                0611  1479          ; lock that is blocked by the current lock, then that lock
                                0611  1480          ; and all the following locks are blocked. For locks that
                                0611  1481          ; are on the granted queue we do not maximize the granted
                                0611  1482          ; mode, for all other queues we will maximize the request mode.
                                0611  1483          ;
                                0611  1484          ; If this lock is not on the granted queue, we skip right to the
                                0611  1485          ; search of the other queue locks.
                                0611  1486          ;
                                0611  1487          ASSUME  LKB$K_GRANTED  EQ  1
                                0611  1488          ASSUME  LKB$K_CONVERT  EQ  0
                                0611  1489          ASSUME  LKB$K_WAITING  EQ  -1
                                0611  1490          ASSUME  RSB$L_CVTQFL  EQ  RSB$L_GRQFL+8
                                0611  1491          ASSUME  RSB$L_WTQFL  EQ  RSB$L_CVTQFL+8
                                0611  1492
          55   34 A9   9A      0611  1493          MOVZBL  LKB$B_RQMODE(R9),R5     ; Get the current lock's requested mode
              57   59   D0      0615  1494          MOVL    R9,R7                   ; R7 will point to other LKB's
                                0618  1495                                          ;  after the one pointed to by R9
               36 A9   95      0618  1496          TSTB    LKB$B_STATE(R9)         ; Which queue is lock on?
                    21   19     061B  1497          BLSS    20$                     ; Br if waiting
                    22   13     061D  1498          BEQL    30$                     ; Br if converting
                                061F  1499          ;
                                061F  1500          ;   Lock is on the granted queue
                                061F  1501          ;
          55   35 A9   9A      061F  1502          MOVZBL  LKB$B_GRMODE(R9),R5     ; Get the current lock's granted mode
                                0623  1503
          57   38 A7   D0      0623  1504 10$:     MOVL    LKB$L_SQFL(R7),R7       ; Get next lock on state queue
              58   57   D1      0627  1505          CMPL    R7,R8                   ; Reached head of queue yet?
                    5B   13     062A  1506          BEQL    90$                     ; Br if yes
              57   38   C2      062C  1507          SUBL    #LKB$L_SQFL,R7          ; Back up to point at start of LKB
          50   35 A7   9A      062F  1508          MOVZBL  LKB$B_GRMODE(R7),R0     ; Get the lock's granted mode
00000000'EF45   50   E0      0633  1509          BBS     R0,-                    ; Branch if compatible
                         E7     063B  1510                  L^LCK$COMPAT_TBL[R5],10$;
                                063C  1511          ;
                                063C  1512          ; Have an incompatible, return the lock info. for all succeeding locks
                                063C  1513          ;
                    62   11     063C  1514          BRB     120$                    ; Return lock info.
                                063E  1515
                                063E  1516 20$:     ;
                                063E  1517          ; Lock is initially on the waiting queue.
                                063E  1518          ;
               58   08   C0      063E  1519          ADDL    #8,R8                   ; Advance R8 one queue header
                                0641  1520 30$:     ;
                                0641  1521          ; Lock is initially on the converting queue, OR we have
                                0641  1522          ; reached the queue header. Advance R8 to point to the next
                                0641  1523          ; queue header in the RSB.
                                0641  1524          ;
               58   08   C0      0641  1525          ADDL    #8,R8                   ; Advance R8 one queue header
                                0644  1526          ;
                                0644  1527          ; Run thru all locks on either the converting or waiting queue
                                0644  1528          ; lock for any locks blocked by the maxmode in R5.
                                0644  1529          ;
          57   38 A7   D0      0644  1530 40$:     MOVL    LKB$L_SQFL(R7),R7       ; Get next lock in queue
              58   57   D1      0648  1531          CMPL    R7,R8                   ; Reached end of queue?
                    3A   13     064B  1532          BEQL    90$                     ; Br if Yes, all done
              57   38   C2      064D  1533          SUBL    #LKB$L_SQFL,R7          ; Back up to point at start of LKB
          50   34 A7   9A      0650  1534          MOVZBL  LKB$B_RQMODE(R7),R0     ; Get requested mode
              51   55   D0      0654  1535          MOVL    R5,R1                   ; Save old maxmode
```

```
                                E 4
                     0657  1536          ;
                     0657  1537          ; Maximize lock modes (in R0 and R5) and see if this lock (R7) is
                     0657  1538          ; incompatible with (the previous) maxmode.  The maximization function
                     0657  1539          ; is a simple arithmetic maximum except if the two modes are CW and PR.
                     0657  1540          ; In that case the maximum of CW and PR is PW.  PW is incompatible
                     0657  1541          ; with everything either CW or PR is incompatible with.
                     0657  1542          ;
          55 50 91   0657  1543          CMPB    R0,R5                   ; Current mode greater than maxmode?
             20 13   065A  1544          BEQL    80$                     ; Br if No, they're equal
             0C 1A   065C  1545          BGTRU   50$                     ; Br if Yes, compute new maxmode
       02 50 91      065E  1546          CMPB    R0,#LCK$K_CWMODE        ; Br if No, is current mode CW?
             19 12   0661  1547          BNEQ    80$                     ; Br if No, maxmode = R2
       03 55 91      0663  1548          CMPB    R5,#LCK$K_PRMODE        ; Br if Yes, is maxmode PR?
             14 12   0666  1549          BNEQ    80$                     ; Br if No, maxmode = R2
             0A 11   0668  1550          BRB     60$                     ; Br if Yes, new maxmode is PW
       02 55 91      066A  1551 50$:     CMPB    R5,#LCK$K_CWMODE        ; Is maxmode CW?
             0A 12   066D  1552          BNEQ    70$                     ; Br if No, maxmode = R0
       03 50 91      066F  1553          CMPB    R0,#LCK$K_PRMODE        ; Br if Yes, is current mode PR?
             05 12   0672  1554          BNEQ    70$                     ; Br if No, maxmode = R0
          55 04 90   0674  1555 60$:     MOVB    #LCK$K_PWMODE,R5        ; Have CW and PR; maxmode = PW
             03 11   0677  1556          BRB     80$
          55 50 90   0679  1557 70$:     MOVB    R0,R5                   ; Maxmode = R0
                     067C  1558
00000000'EF41 50 E1   067C  1559 80$:     BBC     R0,-                    ; Branch if incompatible
             1B       0684  1560                  L^LCK$COMPAT_TBL[R1],120$;  with saved maxmode
             BD 11    0685  1561          BRB     40$                     ; Else, check next lock in RSB
                      0687  1562
          58 08 C0   0687  1563 90$:     ADDL    #8,R8                   ; Advance R8 one queue header
       57 C8 A8 9E   068A  1564          MOVAB   -LKB$L_SQFL(R8),R7      ; Prepare to process that queue
             28 C1   068E  1565          ADDL3   #RSB$L_WTQFL+8,-        ; Get address past waiting queue
    50 50 A9         0690  1566                  LKB$L_RSB(R9),R0
          50 58 D1   0693  1567          CMPL    R8,R0                   ; Have we done all the queues?
             AC 12   0696  1568          BNEQ    40$                     ; Br if Not, repeat for remaining queue
                     0698  1569
          50 01 9A   0698  1570 100$:    MOVZBL  #1,R0                   ; Success indicator
       0066 8F BA    069B  1571          POPR    #^M<R1,R2,R5,R6>        ; Restore registers
                05   069F  1572          RSB
                     06A0  1573
                     06A0  1574          ;
                     06A0  1575          ; We have found the first incompatible lock
                     06A0  1576          ;
          56 18 C2   06A0  1577 120$:    SUBL    #LKI$C_LENGTH,R6        ; Any room left in buffer?
             F3 19   06A3  1578          BLSS    100$                    ; Br if not
          FD83 30    06A5  1579          BSBW    LOCK_INFO               ; Else, return lock info.
       57 38 A7 D0   06A8  1580 130$:    MOVL    LKB$L_SQFL(R7),R7       ; Get next lock in queue
          58 57 D1   06AC  1581          CMPL    R7,R8                   ; Reached end of queue?
             05 13   06AF  1582          BEQL    140$                    ; Br if Yes, skip to next queue
          57 38 C2   06B1  1583          SUBL    #LKB$L_SQFL,R7          ; Back up to point at start of LKB
             EA 11   06B4  1584          BRB     120$                    ; Return the lock info.
                     06B6  1585
          58 08 C0   06B6  1586 140$:    ADDL    #8,R8                   ; Advance R8 one queue header
       57 C8 A8 9E   06B9  1587          MOVAB   -LKB$L_SQFL(R8),R7      ; Prepare to process that queue
             28 C1   06BD  1588          ADDL3   #RSB$L_WTQFL+8,-        ; Get address past end of queues
    50 50 A9         06BF  1589                  LKB$L_RSB(R9),R0
          50 58 D1   06C2  1590          CMPL    R8,R0                   ; Have we done all queues?
             D1 13   06C5  1591          BEQL    100$                    ; Br if Yes, leave
             DF 11   06C7  1592          BRB     130$                    ; Else, loop thru remaining queues
```

06C9 1593

```
                                        .SBTTL  LKI_ALLOCATE - Allocate a system buffer

                                ;++
                                ; FUNCTIONAL DESCRIPTION:
                                ;
                                ;       This routine attempts to allocate a system buffer and intialize
                                ;       the structure type.
                                ;
                                ; CALLING SEQUENCE:
                                ;
                                ;       JSB/BSB
                                ;
                                ; INPUTS:
                                ;
                                ;       R6      Size of desired buffer minus header
                                ;
                                ; IMPLICIT INPUTS:
                                ;
                                ;       IPL = IPL$_SYNCH
                                ;
                                ; OUTPUTS:
                                ;
                                ;       R0      Completion status for request
                                ;       R2      Address of the system buffer at data portion of buffer
                                ;       R10     Address of start of the system buffer
                                ;
                                ; SIDE EFFECTS:
                                ;
                                ;       none
                                ;--

                                LKI_ALLOCATE:
                        1A BB   PUSHR   #^M<R1,R3,R4>           ; Save registers
54   00000000'EF        D0      MOVL    SCH$GL_CURPCB,R4        ; Get PCB address
     51   56   0C       C1      ADDL3   #12,R6,R1               ; Compute size of system buffer
                                ;
                                ; NOTE: The exec routine EXE$BUFFRQUOTA cannot be called, since
                                ; it will lower IPL and destroy all synchronization.
                                ;
50   00000000'EF        3C      MOVZWL  IOC$GW_MAXBUF,R0        ; Get maximum buffer size allowed
          50   51       D1      CMPL    R1,R0                   ; Is buffer too big?
               28       1A      BGTRU   20$                     ; Br if yes, error
     50   0080 C4       D0      MOVL    PCB$L_JIB(R4),R0        ; Get JIB address
       24 A0   51       D1      CMPL    R1,JIB$L_BYTLM(R0)      ; Is BYTLM quota okay?
               1D       1A      BGTRU   20$                     ; Br if not, error
       20 A0   51       D1      CMPL    R1,JIB$L_BYTCNT(R0)     ; Is BYTCNT quota okay?
               17       1A      BGTRU   20$                     ; Br if not, error
     00000000'EF        16      JSB     EXE$ALONONPAGED         ; Try and allocate a buffer
          13   50       E9      BLBC    R0,30$                  ; Br if failed
          5A   52       D0      MOVL    R2,R10                  ; Set address of buffer to deallocate
                                ;
                                ; Initialize structure header
                                ;
               82       7C      CLRQ    (R2)+                   ; Zero return size, unused fields
          82   51       B0      MOVW    R1,(R2)+                ; Set structure size
          82   13       B0      MOVW    #DYN$C_BUFIO,(R2)+      ; Set structure type
               1A       BA      POPR    #^M<R1,R3,R4>           ; Restore registers
```

```
                                                H 4
                     05  0709  1652          RSB
                         070A  1653
          50   1C    3C  070A  1654 20$:     MOVZWL  #SS$_EXQUOTA,R0         ; Set error return
               F8    11  070D  1655          BRB     10$                     ; Return to caller
                         070F  1656
     50   0124  8F    3C  070F  1657 30$:     MOVZWL  #SS$_INSFMEM,R0         ; Set error return
               F1    11  0714  1658          BRB     10$                     ; Return to caller
                         0716  1659
                         0716  1660
                         0716  1661          .END
```

```
$$T1                            = 00000000
ACB$L_KAST                      = 00000018
ACB_L_COUNT                       0000002C
ACB_L_DADDR                       0000001C
ACB_L_EFN                         00000020
ACB_L_ILIST                       00000030
ACB_L_IOSB                        00000024
ACB_L_OPID                        00000028
ASTADR                          = 00000014
ASTPRM                          = 00000018
BSTRING                         = 00000001
CHECKITEM                         000001F3 R      02
CHECK_SPC                         000002AD R      02
CLU$GL_CLUB                       ********   X    02
CLUB$L_LOCAL_CSID               = 00000060
CSTRING                         = 00000002
DYN$C_BUFIO                     = 00000013
EFN                             = 00000004
EXE$ALONONPAGED                   ********   X    02
EXE$DEANONPAGED                   ********   X    02
EXE$GETLKI                        00000000 RG     03
EXE$IPID_TO_EPID                  ********   X    02
EXE$PROBEW                        ********   X    02
EXE_GETLKI                        00000098 R      02
GETLKB                            00000493 R      02
GET_REMLKI                        000001D0 R      02
GRET                              00000177 R      02
IOC$GW_MAXBUF                     ********   X    02
IOSB                            = 00000010
IPL$_ASTDEL                     = 00000002
IPL$_SYNCH                      = 00000008
ITMLST                          = 0000000C
JIB$L_BYTCNT                    = 00000020
JIB$L_BYTLM                     = 00000024
LCK$CHECK_STALL                   ********   X    02
LCK$COMPAT_TBL                    ********   X    02
LCK$GL_IDTBL                      ********   X    02
LCK$GL_MAXID                      ********   X    02
LCK$K_CWMODE                    = 00000002
LCK$K_PRMODE                    = 00000003
LCK$K_PWMODE                    = 00000004
LIMSG$K_ZERO                    = 00000000
LIMSG$L_LCKCOUNT                = 0000002C
LIMSG$L_RSBREFCNT               = 00000028
LIMSG$L_STATE                   = 00000024
LIMSG$Q_VALBLK                  = 00000030
LKB$B_GRMODE                    = 00000035
LKB$B_RQMODE                    = 00000034
LKB$B_STATE                     = 00000036
LKB$K_CONVERT                   = 00000000
LKB$K_GRANTED                   = 00000001
LKB$K_WAITING                   = FFFFFFFF
LKB$L_CSID                      = 00000058
LKB$L_EPID                      = 00000014
LKB$L_LKID                      = 00000030
LKB$L_PARENT                    = 00000048
LKB$L_PID                       = 0000000C
LKB$L_REMLKID                   = 00000054
LKB$L_RSB                       = 00000050
LKB$L_SQBL                      = 0000003C
LKB$L_SQFL                      = 00000038
LKB$S_MODE                      = 00000002
LKB$V_MODE                      = 00000000
LKB$V_MSTCPY                    = 00000004
LKB$W_REFCNT                    = 0000004C
LKB$W_STATUS                    = 0000002A
LKBTBL                            00000002 R      02
LKI$C_LENGTH                    = 00000018
LKI$C_LKBTYPE                   = 00000001
LKI$C_RSBTYPE                   = 00000002
LKI$C_WAITING                   = FFFFFFFF
LKI$SEARCH_BLOCKEDBY              0000060D RG     02
LKI$SEARCH_BLOCKING               0000056C RG     02
LKI$SND_BLKBY                     ********   X    02
LKI$SND_BLKING                    ********   X    02
LKI$SND_LOCKS                     ********   X    02
LKI$SND_STDREQ                    ********   X    02
LKI$V_SYSNAM                    = 0000001F
LKI$_BLOCKEDBY                  = 00000206
LKI$_BLOCKING                   = 00000207
LKI$_LASTLKB                    = 00000106
LKI$_LASTRSB                    = 00000209
LKI$_LCKCOUNT                   = 00000205
LKI$_LCKREFCNT                  = 00000103
LKI$_LOCKID                     = 00000104
LKI$_LOCKS                      = 00000208
LKI$_NAMSPACE                   = 00000200
LKI$_PARENT                     = 00000102
LKI$_PID                        = 00000100
LKI$_REMLKID                    = 00000105
LKI$_RESNAM                     = 00000201
LKI$_RSBREFCNT                  = 00000202
LKI$_STATE                      = 00000101
LKI$_SYSTEM                     = 00000204
LKI$_VALBLK                     = 00000203
LKID                            = 00000008
LKI_ALLOCATE                      000006C9 R      02
LOCAL_SPACE                     = FFFFFFF8
LOCK_INFO                         0000042B R      02
MAXCOUNT                          00000000 R      02
MAXSTRUC                        = 00000002
MAX_LKB_ITEM                    = 00000005
MAX_RSB_ITEM                    = 00000008
MOVEIT                            0000025A R      02
PCB$L_JIB                       = 00000080
PCB$L_PID                       = 00000060
PCB$Q_PRIV                      = 00000084
PCB$W_ASTCNT                    = 00000038
PCB$W_GRP                       = 000000BE
PR$_IPL                         = 00000012
PRV$V_SYSLCK                    = 0000001E
PRV$V_WORLD                     = 00000010
PSL$C_EXEC                      = 00000001
PSL$C_KERNEL                    = 00000000
```

```
PSL$S_PRVMOD                    = 00000002
PSL$V_PRVMOD                    = 00000016
RESERV                          = 0000001C
RSB$B_RMOD                      = 0000004E
RSB$B_RSNLEN                    = 0000004F
RSB$L_CSID                      = 00000038
RSB$L_CVTQFL                    = 00000018
RSB$L_GRQFL                     = 00000010
RSB$L_WTQFL                     = 00000020
RSB$Q_VALBLK                    = 00000028
RSB$W_GROUP                     = 0000004C
RSB$W_REFCNT                    = 00000040
RSBTBL                            00000026 R      02
SAVED_IPL                       = FFFFFFFC
SCH$CREF                          ********   X    02
SCH$GL_CURPCB                     ********   X    02
SCH$POSTEF                        ********   X    02
SPC_BLOCKEDBY                     0000035D R      02
SPC_BLOCKING                      00000392 R      02
SPC_LCKCOUNT                      00000332 R      02
SPC_LOCKS                         000003C7 R      02
SPC_NAMSPACE                      00000321 R      02
SPC_PARENT                        000002FD R      02
SPC_PID                           000002D2 R      02
SPC_REMLKID                       00000347 R      02
SPC_STATE                         000002EB R      02
SPC_SYSTEM                        0000030C R      02
SPECIAL                           0000005C R      02
SPECIAL_LEN                     = 0000000A
SS$_ACCVIO                      = 0000000C
SS$_BADPARAM                    = 00000014
SS$_BUFFEROVF                   = 00000601
SS$_EXQUOTA                     = 0000001C
SS$_INSFMEM                     = 00000124
SS$_IVLOCKID                    = 00002124
SS$_IVMODE                      = 00000354
SS$_NOMORELOCK                  = 00000A08
SS$_NORMAL                      = 00000001
SS$_NOSYSLCK                    = 000028F4
SS$_NOWORLD                     = 00002884
SYS$DCLAST                        ********  GX    02
VALUE                           = 00000000
VERIFYLOCKID                      000004E4 R      02
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000030 (48.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
WSYSGETLKI	00000716 (1814.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
YEXEPAGED	00000008 (8.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	34	00:00:00.07	00:00:00.38
Command processing	127	00:00:00.62	00:00:06.40
Pass 1	488	00:00:19.74	00:00:53.40
Symbol table sort	0	00:00:02.90	00:00:08.60
Pass 2	289	00:00:05.01	00:00:11.37
Symbol table output	19	00:00:00.16	00:00:00.37
Psect synopsis output	2	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	961	00:00:28.53	00:01:20.55

The working set limit was 2100 pages.
113868 bytes (223 pages) of virtual memory were used to buffer the intermediate code.
There were 100 pages of symbol table space allocated to hold 1809 non-local and 102 local symbols.
1661 source lines were read in Pass 1, producing 22 object records in Pass 2.
39 pages of virtual memory were used to define 38 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SHRLIB]CLUSTER.MLB;1	1
_255DUA28:[SYS.OBJ]LIB.MLB;1	18
_255DUA28:[SYSLIB]STARLET.MLB;2	13
TOTALS (all libraries)	32

1957 GETS were required to define 32 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:SYSGETLKI/OBJ=OBJ$:SYSGETLKI MSRC$:SYSGETLKI/UPDATE=(ENH$:SYSGETLKI)+EXECML$/LIB+SHRLIB$:CLUSTER/LIB
```

SYSGETSYI
LIS

SYSGETPTI
LIS

SYSGETTIM
LIS

SYSIMGFIX
LIS

SYSGETLKI
LIS

SYSGETMSG
LIS

SYSIMGACT
LIS